REPORT T-104 FEBRUARY, 1981

CSL COORDINATED SCIENCE LABORATORY

ARTIFICIAL INTELLIGENCE AND HUMAN ERROR PREVENTION: A COMPUTER AIDED DECISION MAKING APPROACH

TECHNICAL REPORT No. 4

SURVEY AND ANALYSIS OF RESEARCH ON LEARNING SYSTEMS FROM ARTIFICIAL INTELLIGENCE

BY:
R.T. CHIEN
W.P.-C. HO
D.C. CHEN
S.C. KELLER
R.J. FLETCHER
S.E. CROSS
Y.C. PAN
R.L. YEN

THIS WORK WAS SUPPORTED IN PART BY THE UNITED STATES DEPARTMENT OF TRANSPORTATION UNDER CONTRACT DOT-FA79WA-4360 AB AND THE FEDERAL AVIATION ADMINISTRATION

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

Coordinated Science Laboratory

Advanced Automation Group

Survey and Analysis of Research on

- - -

Learning Systems from Artificial Intelligence

R. T. Chien
W. P.-C. Ho
D. C. Chen
S. C. Keller
R. J. Fletcher
S. E. Cross
Y. C. Pan
R. L. Yen

February 5, 1981

Table of Contents

Section I

Introduction

The world of Air Traffic Control (ATC) has become more complex and more hectic. Increasing reliance on aviation in all sectors has inspired burgeoning development in aircraft sophistication and rapid growth in aircraft number. These trends have combined to increase the very heavy, real-time burden on the ATC specialist, who is in urgent need of relief. The unequivocal answer to this need is to incorporate computer assistance. What must be resolved is the form that this automation is to take.

The current state of computer assistance in air traffic control is fairly basic. Computers are used for high-reliability data processing of flight and radar data to assist the ATC specialist. The specialists are still responsible for all the work in providing safe and efficient air traffic control. Computers play no part in the higher level jobs of planning strategy or implementing tactics. In the light of rapid and fruitful development of artificial intelligence research, much more can be done to provide higher levels of assistance to the ATC specialist (specifically in strategy and tactics skills).

A research project which addresses the problem of providing sophisticated and intelligent assistance in the ATC domain is under investigation at the Coordinated Science Laboratory at the University of Illinois. The focus of research efforts is the identification of conceptual knowledge, and the definition of conceptual learning structures needed for defining the architecture of an expert-level system which improves its performance by learning. This skill acquisition is envisioned to

take place in the areas of planning and problem-solving through systematic study of examples in the air traffic domain. System performance will evolve by inducing and employing the underlying heuristic strategies which the ATC specialist uses.

As a part of the research effort, a comprehensive survey and analysis of the leading edge of research in learning was undertaken. The result of that study are summarized in this report.

During the study, we paid special attention to learning approaches in the planning and problem-solving domains. We also paid special attention to the domain complexity and the initial knowledgeability of the system. Three cohesive approaches which reflect a continuity in effort emerged from this study of the state of the art in learning. These approaches, at Stanford University, M. I. T., and Carnegie-Mellon University, are summarized and analyzed in the three following sections. Papers which resulted from significant work at these universities are individually analyzed. A cohesive analysis summarizing the underlying approach concludes each of these three sections. Significant work has been done at various other research institutions, but on a smaller scale of research concentration. A last section covers some of these many other interesting papers which have been produced in the field of learning.

While some interesting results has been obtained, the area of learning by an expert system in a complex planning and problem-solving domain remains underdeveloped.

Section II

Stanford University

Heuristic Programming Project

A Brief Introduction to the Stanford University Heuristic Programming Project Approach

Knowledge engineering is defined by E. Feigenbaum as "the art of bringing the principles and tools of AI research to bear on difficult applications problems requiring experts' knowledge for their solution". Researchers at the Stanford University Heuristic Programming Project (H. P. P.) have defined a system architecture and a corresponding design methodology for constructing expert systems. This represents one approach to knowledge engineering.

The H. P. P. perspective on learning developed out of the difficulties they encountered in trying to carry out the design concepts of this approach. Therefore, we first take a brief look at the H. P. P. approach.

I Foundations:

In many domains, such as medicine, there appears to be no cohesive, well-structured theory built on first principles. How do the acknowledged experts do what they do in these domains? First of all, the expert's specialist knowledge seems to be a collection of heuristics gleaned from long experience. These heuristics are based on inexact knowledge formulated into judgmental rules. Second of all, the expert can use them to make "good guesses", and yet he finds it difficult to articulate how he did what he did, and why. The expert doesn't seem to know

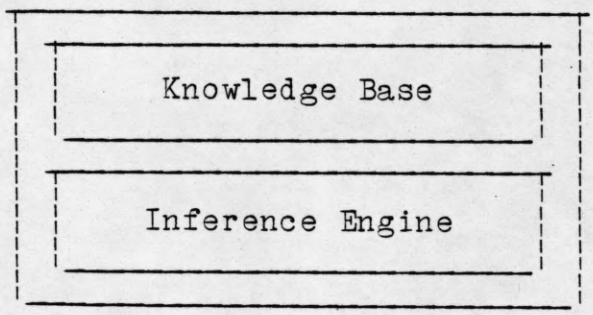
what he knows. Third of all, the expert's specialist knowledge is indisputably vast. Yet it is at a fairly shallow level. This is a manifestation of the absence of a unified domain theory. Nonetheless, the expert performs at a level to merit his title. Surface knowledge appears to be sufficient. H. P. P. researchers drew these conclusions from their observations of experts in action. Furthermore, they showed a research preference for construction of intelligent artifacts over modeling of human cognition. These factors shaped the H. P. P. approach.

II Design Scenario:

H. P. P. expert system design is implemented by a design team of two members, and can be decomposed into three functional phases. The first team member is the domain expert (there may actually be many experts who rotate), who is the source of specialist knowledge. The second team member is the knowledge engineer, who translates that knowledge into an expert system within the H. P. P. architecture. The first design phase is knowledge extraction, a difficult task which is the bottleneck in expert system realization. The knowledge engineer works with the domain expert by going over numerous case studies to alleviate his difficulty in articulating his expertise. From his specific responses to similar cases, the design team attempts to encapsulate the general reasoning the expert employs. The second design phase is knowledge base construction. The design team

determines the level and amount of knowledge in each judgmental rule. This phase is actually interleaved with that of knowledge extraction. The last design phase is system validation. This system test is performed at various stages of knowledge base completion to verify that the direction of system development is satisfactory to the domain expert. The expert system performance on test cases is compared to human expert performance to determine knowledge base sufficiency and accuracy.

III System Architecture:



Knowledge Base - The knowledge base is composed of production rules. Each rule is of the form <premise> => <action>. The <premise> is a conjunctive list of conditions which, if satisfied, will trigger the conclusion in the <action>. Essentially, each rule captures an independent, complete chunk of expert reasoning. For example, Rule 027 below, part of an investment expert system, will conclude that AT&T is a good investment if three conditions are satisfied:

Rule 027

If 1) the time scale of the investment is long-term, and
2) the desired return of the investment is greater than 10%, and
3) the area of the investment is not known

Then AT&T is a likely (.4) choice for the investment

Inference Engine - The inference engine performs goal-directed, backward-chaining of production rules. Starting from a final goal <action> (eg. recommend AT&T), the inference engine makes a subgoal of each corresponding <premise> condition. The inference engine then attempts to find additional rules whose <actions> match the new subgoals (eg. look for rules that conclude that time scale is long-term). These rules are chained onto the inference tree. Their <premise>'s are made into subgoals and so on. Essentially, this control structure is attempting to justify the application of a rule by attempting to satisfy the conditions of that rule. This satisfaction may be provided directly by the user, or it may be the conclusion of other rules whose conditions must be satisfied in turn. This inference-chaining backwards from the goal is reminiscent of human reasoning. The inference engine performs exhaustive depth-first search. The leaf nodes of the resultant inference tree are invariably pieces of input domain data.

The H. P. P. architecture is a consequence of observing experts practicing their art. The source of the expert's performance is his vast collection of surface knowledge. The

domain is knowledge-rich but at a shallow level. Thus, the architecture configures the expert system to draw its power from the rich, domain-specific knowledge base applied thru a simple, domain-independent inference engine.

This architecture also facilitates explaining system reasoning to users. Each rule in the knowledge base is a chunk of domain-specific knowledge encoded in the format experts use to rationalize a conclusion. When the inference engine completes an inference chain of rules from the goal to user input data, it has constructed a very human line-of-reasoning. This is the key to the explanation capability of the expert system in the performance phase. This line-of-reasoning is an artifact of the way knowledge is articulated by the domain expert and used in system performance.

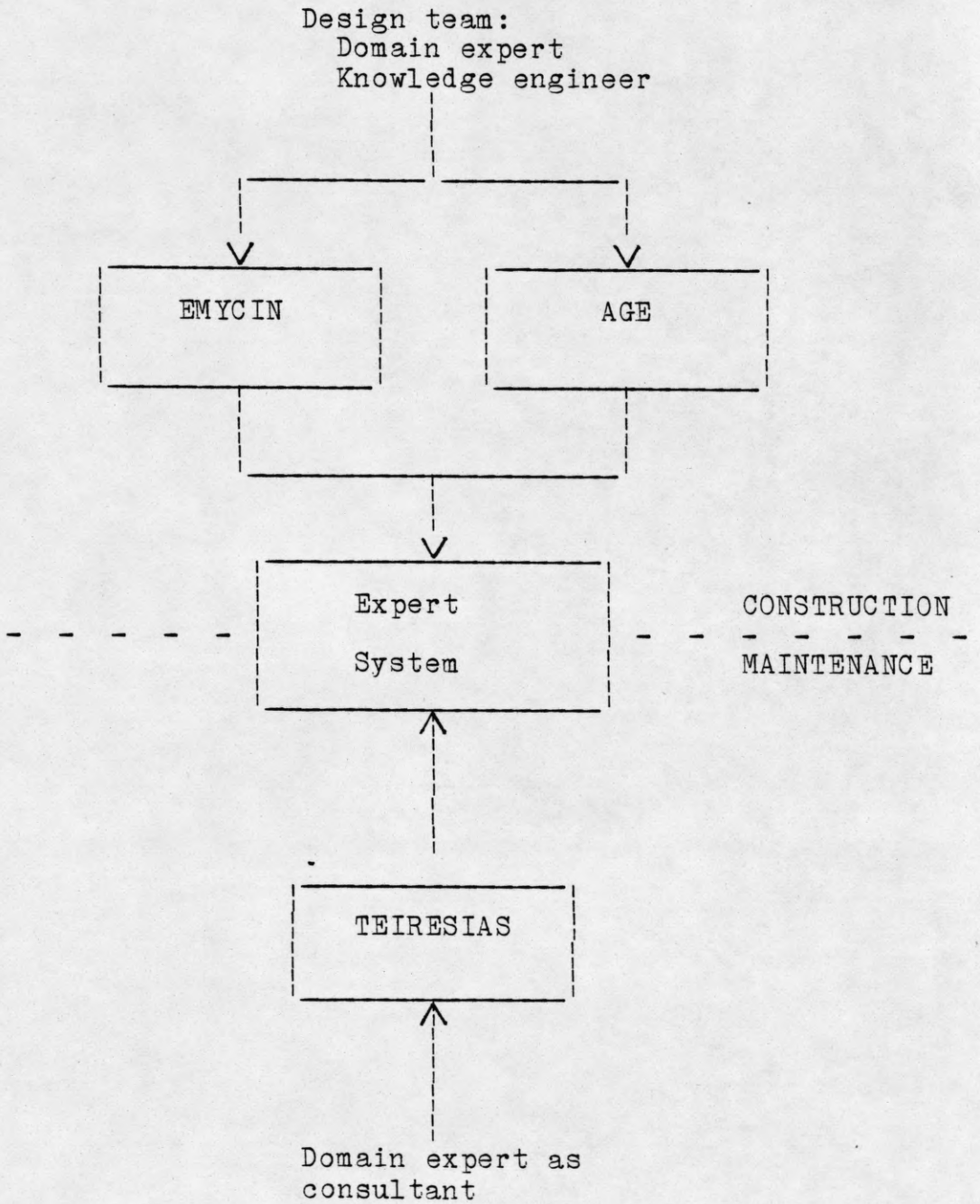
IV Overview of H. P. P. Learning Systems:

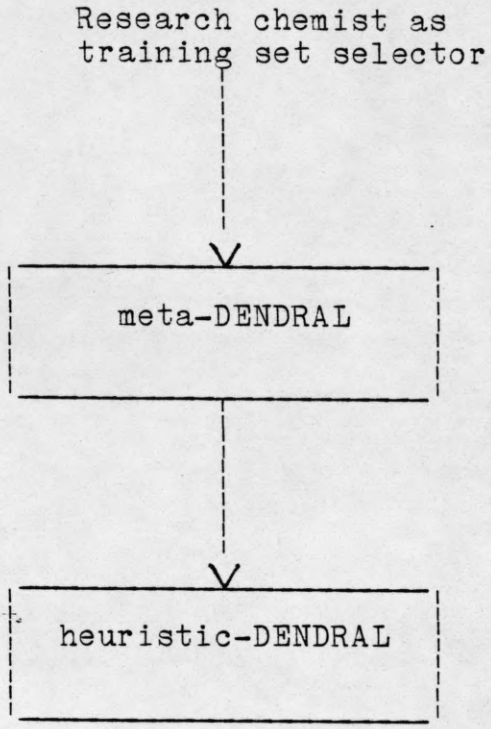
Knowledge acquisition or learning occurs either in the knowledge base construction context, or in the knowledge base maintenance context. In the construction context, the H. P. P. systems which aid the design team provide the design environment. The target expert system, within this environment, is "learning by being told". EMYCIN and AGE are both attempts to provide knowledge engineering tools which raise the programming task of encoding extracted rules to a higher level. In the maintenance context, the knowledge engineer is no longer needed. The H. P. P. system, TEIRESIAS, acts as an intelligent interface

between the mature expert system and the consultant domain expert. It learns by understanding the expert system knowledge base and thereby taking on the role of the knowledge engineer. This understanding is based on the concept of meta-knowledge, knowledge about knowledge. TEIRESIAS learns by knowing what to ask.

These three systems are general to the H. P. P. architecture. Any design and maintenance of an expert system within that architecture could be simplified by employing these systems. Meta-DENDRAL, on the other hand, is a learning system designed with one expert system in mind, heuristic DENDRAL. It takes advantage of the domain characteristics to achieve expert learning performance. Meta-DENDRAL learns by induction. This learning is based on the concept of version spaces, a candidate elimination approach to rule learning.

Stanford University Heuristic Programming Project
Learning Systems




Research chemist as
training set selector
meta-DENDRAL
heuristic-DENDRAL

Papers: A Model for Learning Systems

Reid Smith, Tom Mitchell, Richard Chestek,
and Bruce Buchanan
IJCAI, 1977.

Models of Learning Systems

Bruce Buchanan, Tom Mitchell, Reid Smith,
and C. Richard Johnson Jr.
Stanford University Technical Report,
January 1979.

I Problem:

Learning systems have been the subject of wide research interest in many different scientific communities for a number of years. Since each community views learning systems from its own perspective and has developed its own jargon for the various facets of learning system research, it is often difficult for members of these communities to recognize that problems which appear unrelated as a result of variations in terminology may in fact be identical. A learning system model is presented in this paper which provides a common language for systems constructed from different perspectives. This model encourages examination of the strengths and weaknesses of the individual functional components necessary for any learning system.

II Approach:

A learning system is defined to be any system which uses information obtained during one interaction with its environment to improve its performance during future interactions. Research

in learning system can be roughly divided into two approaches. The first approach centers on the concept of an adaptive system and is primarily associated with research in pattern recognition and control theory. The second approach centers on knowledge structure and is associated with artificial intelligence.

The first approach views learning as successive approximation of the unknown parameters of a mathematical structure that represents the system under study. The emphasis has been on parameter adjustment and the achievement of stable, reliable performance. Problems are commonly formulated in stochastic terms. Statistical procedures are used to achieve optimal performance with respect to some performance criterion such as the probability of correct pattern classification or the mean square error.

The second approach differs from the first in that whereas the pattern recognition and control research emphasizes adjustment of parameters, AI research emphasizes construction of symbolic structures that represent conceptual relations. Another difference is that the AI approach believes a learning system should have sufficient internal structure such that not only does it have the capacity for high performance, but it can also explain its performance in symbolic terms.

The environment in which a learning system operates has a strong influence on its design. Basically, the learning system environments can be categorized as supervised learning or unsupervised learning. A supervised learning environment

provides the correct response for each training instance, while an unsupervised learning environment does not. In other words, supervised learning systems operate within a stimulus-response environment in which the desired learning system output is supplied with each training instance. Unsupervised learning systems operate within an environment of instances for which the correct response is not directly available. While many factors influence the choice of the environment, the proposed learning system model, however, is applicable to systems operating in either environment.

2.1 The Proposed Learning Model

The learning system model described in this paper is a functional decomposition of the components necessary to a learning system. The model is shown in Figure 1. The components of the model are conceptual entities which specify functions that must be performed during learning. Although the functional decomposition does not necessarily imply the physical decomposition of a system, a one-to-one correspondence would provide conceptual clarity and functional modularity.

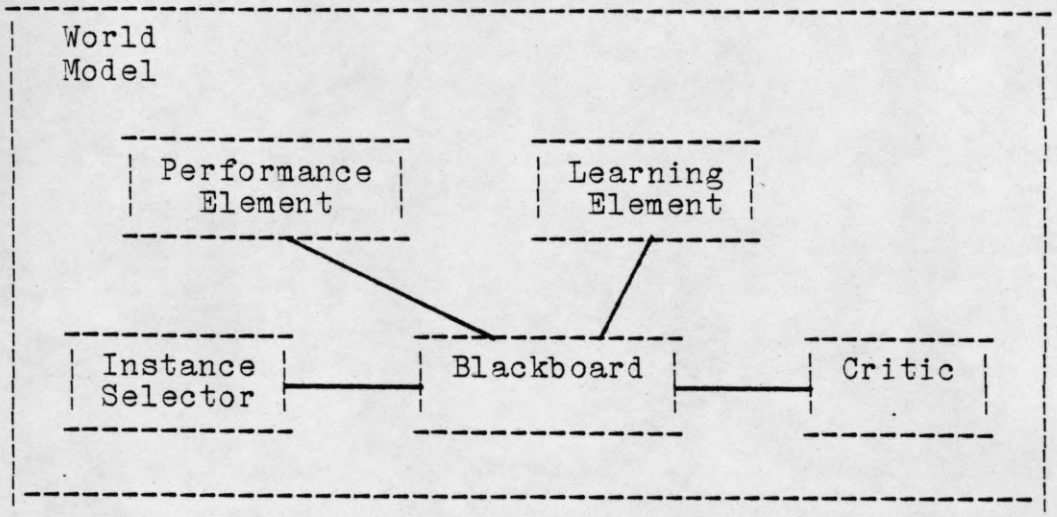


Figure 1. The Components of the Learning System Model

2.1.1 The Performance Element

The performance element performs the stated task of the system. It does what the system would do if the system does not have to learn. The performance element is responsible for generating an output in response to a training instance. In any learning system the ability to improve performance presupposes a method of communicating learned information to the performance element. Thus the performance element should be constructed so that its knowledge base and control knowledge are readily available to the other system components. The performance element of existing systems also vary in the richness in which they may be extended by learning. For example, systems whose operation is determined by a set of production rules have the potential to exhibit richer variations than systems whose operation is keyed only to the adjustment of parameter values.

2.1.2 The Instance Selector

The instance selector selects the suitable training instances to be used by the learning system. The instances can be selected with great care by a teacher or they can be given without regard to the order of presentation. For example, a teacher may choose with great care instances of flawed arches to illustrate the criteria of an arch, while two learning checker-playing systems may increase their performance by testing out their own stategies on each other. When the instances are selected by an external agent, the instance selector is said to be passive. When the instances are generated by an internal process, the instance selector is said to be active.

2.1.3 The Critic

The critic analyzes the output of the performance element in terms of some standard of performance. It may play three roles: evaluator, diagnostician, or therapist. The critic always operates as an evaluator in that it assesses the behavior of the performance element. In addition to being an evaluator the critic may also operate as a diagnostician. In which case he also localizes the reason for poor performance. Finally, the critic may operate as a therapist and make specific recommendations for improvement or suggestions about future instances.

In some learning systems the functions of the critic have been left to the human who use the system. For example, MYCIN/TEIRESIAS employs a human critic, acting as evaluator, diagnostician, and therapist to suggest modifications to its rule base.

2.1.4 The Learning Element

The learning element is an interface between the critic and the performance element. It takes the critic's analysis and makes specific changes to the system. The method incorporating newly learned information is dependent on the system's representation of knowledge and may vary even for systems using similar representations.

2.1.5 The Blackboard

The blackboard is a global data base which also functions as a system communications mechanism. The blackboard is accessible to all the functional components so that system information is available to all the modules. The blackboard contains the knowledge base as well as temporary information used by the learning system components.

2.1.6 The World Model

The world model contains the conceptual framework within which the system operates, i.e. it knows the legal moves of the domain. The world model contains definitions of objects and

relations in the task domain, the syntax and semantics of the information to be learned, and the methods to be used by the learning system. Where possible, world model constraints should be made explicit in order to allow easy modification during the design process.

III Contributions and Discussion:

The learning system model proposed in this paper provides a common language for characterizing and comparing different kinds of learning systems. The model is also useful as a conceptual guide for learning system design because it isolates the essential functional components and the information that must be available to these components. In addition to the proposed model, this paper also suggests that the physical modules of a learning system should correspond to the functional modules proposed by this model. This gains transparency and modularity. The spirit of modularity should also extend to the knowledge base--the knowledge should be explicit and accessible. This is especially true for the parts of the learning system that are adjusted in response to learning.

Paper: AGE: A Knowledge-Based Program for Building Knowledge-Based Programs

By: H. Penny Nii and Nelleke Aiello
IJCAI, 1979.

I Problem:

The knowledge-based expert systems developed thus far in artificial intelligence have required close interaction between the computer system expert, the knowledge engineer, and the domain expert. This close interaction makes it difficult for the domain expert to investigate and experiment with different intelligent system architectures. The AGE project is designed to assist the domain expert in constructing knowledge-based expert systems.

II Approach:

The goal of the AGE project is to clarify and make explicit the art of knowledge engineering. The AGE project attempts to formulate the knowledge that the knowledge engineers use in constructing knowledge-based programs and make it available to others in the form of a software laboratory. The final product is to be a collection of building-block programs and an intelligent front-end that will assist the user in constructing knowledge-based programs. It is hoped that the AGE program will speed up the process of building knowledge-based programs and facilitate the dissemination of AI techniques. This is done by packaging common AI software tools so that they need not be

reprogrammed for every problem and by easing the way for those who are not knowledge engineering specialists to write knowledge-based programs.

The AGE program is being developed along two separate fronts. The first is the development of tools to help the user build a variety of knowledge-based programs. The second is the development of intelligence in the interaction between the user and the AGE program.

The building-block components currently available to the users form the blackboard model. The blackboard model consists of three major components. The first component is the blackboard, and it is used as a means of communication and interaction among the knowledge sources, the second major component in the system. The knowledge sources contain the knowledge of the task domain. Presently the knowledge sources are represented as sets of production rules. The third component of the blackboard model is the control component. The control component contains mechanisms that allow the user to specify the conditions for the invocation of the knowledge sources. The control component also selects the items on the blackboard for the focusing of attention.

AGE does not assume that the user is familiar with the options available to him, and it is the responsibility of the intelligent front end to assist the user in interacting with AGE. A tutor subsystem allows the user to browse through a textual knowledge base that contains a description of the building-block

components, a description of how these components are to be used, a description of how they can be constructed, and various illustrative examples. The design subsystem is another component of the front-end that guides the user in the design and the construction of his expert systems. The knowledge necessary for AGE to accomplish this task is represented in a data structure in the form of an AND/OR tree that represents the available structures and the decisions the user must make to design his expert system.

III Contributions and Discussion:

The AGE program is aimed at reducing the overhead involved in constructing a knowledge-based expert system. Thus AGE is an example of "learning by being programmed". The short-term goal of AGE is to provide software tools that a knowledge engineer can use to build knowledge-based programs. The long-term goal of AGE is to assist the domain experts, who are understandably less knowledgeable about the available AI methods, in constructing knowledge-based programs.

Paper: A Domain-Independent Production-Rule System for Consultation Programs

By: William Van Melle
IJCAI, 1979.

I Problem:

Much of the work in artificial intelligence involves the development of computer programs that aid experts in complex reasoning tasks. Examples of such knowledge-based expert systems are the MYCIN program, a consultation program for the diagnosis of and the therapy for patients with infectious diseases, and the PROSPECTOR program, a consultation system for mineral exploration. Such expert systems require large amounts of task-specific knowledge, and experience has shown that building the knowledge base can be very time-consuming. EMYCIN is a system developed at the Heuristic Programming Project at Stanford University to meet this problem. EMYCIN is a domain-independent production-rule system designed for developing expert consultation programs. EMYCIN enhances the knowledge acquisition process and can be considered to be a system that facilitates "learning by being told".

II Approach:

EMYCIN is designed to be the underlying machinery used to construct consultation programs. It is essentially an embellished framework of the MYCIN program. EMYCIN's architecture requires the knowledge base to consist of production

rules operating on associative triples. The control structure is also restricted to a goal-directed backward-chaining of the production rules.

The domain knowledge is encoded as production rules. Each rule has a premise, which is a conjunction of predicates over triples in the knowledge base. If the premise is true, the conclusion in the action part of the rule is drawn. A production rule can be used for inference when used in the forward direction or deduction when used in the backward direction.

EMYCIN supports several features that ease the acquisition of the production rules. EMYCIN's explanation program allows the user to interrogate the system's knowledge, either to find out about inferences made in the particular case at hand or to examine the static knowledge base in general. The explanation program functions as a high-level debugging aid for the system developer. Without having to resort to Lisp-level manipulation, the system developer can examine any inferences that were made, find out why others failed, and thereby correct errors or omissions in the knowledge base.

EMYCIN also supplies a high-level database editor for the data structures in the system. Though the data is entered in its Lisp format, the editor handles all the bookkeeping involved in managing the rule base. The editor also checks for common user errors. In addition, EMYCIN also maintains a library of sample cases. These sample cases are used to test the knowledge base after it has been revised by the expert.

III Contributions and Discussion:

The contribution of EMYCIN is the separation of the domain-independent inference engine and the system support utilities from the domain-specific knowledge base for the production rule consultation systems. Several consultation systems have been written in EMYCIN. The PUFF system diagnoses pulmonary dysfunctions. The HEADMED program diagnoses a range of psychiatric disorders and can recommend drug treatment. SACON is another program built on EMYCIN to provide advice to a structural engineer regarding the use of a large structural analysis program.

Experience with the systems built on EMYCIN hints at several difficulties inherent to such an architecture. The encoding of the many aspects of scientific decision-making in the simple production rule format causes difficulties at times. Similarly, the control structure of the goal-directed backward chaining of rules has also proved to be awkward at times. However, EMYCIN's goal of improving the facilities of acquiring and debugging rules should prove to be useful to systems of "learning by being told".

Paper: Interactive Transfer of Expertise:
Acquisition of New Inference Rules

By: Randall Davis
IJCAI, 1977.

I Problem:

Artificial intelligence expert systems such as DENDRAL and MACSYMA have traditionally been assembled by hand. This tedious construction and maintenance has required a cast of three: (1) The expert system being built, which is the recipient of the vast amount of domain knowledge required for high performance. (2) The domain expert, who is present as the source of that domain expertise but who often knows nothing about programming. (3) The knowledge engineer, who is therefore required to mediate the transfer between expert and computer. This process can be characterized as "learning by being told". TEIRESIAS is an attempt to upgrade this level of learning. It is an expert system which acts as an intelligent assistant to replace the knowledge engineer as intermediary. As such, TEIRESIAS inherits the responsibility of taking part in an interactive, mixed-initiative transfer of knowledge.

II Approach:

The scenario of learning involving TEIRESIAS begins with the consultant. In his role as domain expert, he challenges the expert system with a new problem and tracks its performance. If he is dissatisfied with the solution, TEIRESIAS intervenes to

stage a high-level, mixed-initiative dialogue with him in a restricted subset of natural language. In its role as knowledge engineer, TEIRESIAS guides the domain expert in tracking down the bug. It works with him to correct the faulty chunk of knowledge responsible. If new knowledge is required to remove the bug, it accepts, interprets, and "second-guesses" the contents of the new candidate inference rule. This process continues until both TEIRESIAS and the domain expert are satisfied with the interpretation of the new knowledge. TEIRESIAS helps the domain expert to distill a system-compatible version. TEIRESIAS then integrates this new chunk of knowledge into the expert system's knowledge base.

TEIRESIAS's power lies in its ability to direct the system repair process. It does not ask the domain expert "What should I know about this domain?". Instead, it uses its understanding of the expert system knowledge base, data from the problem-solving session, and responses to previous questions to focus its line-of-questioning. TEIRESIAS builds expectations and thereby knows what to ask.

The key to context-based questioning is the explicit models of knowledge base inference rules and domain data maintained by TEIRESIAS for introspection. These prototypes comprise part of the meta-level knowledge which allows TEIRESIAS to use, examine, abstract, reason about, and direct the application of the expert system's expert knowledge (see section on meta-knowledge). Each prototype provides explicit knowledge of structure and content

for a specific inference rule type or domain data type. The chunk of new knowledge to be understood is matched against a hierarchical graph of prototypes to identify the most appropriate one. This model then provides the framework for guiding context-based questioning. As the expert system knowledge base grows, these models are dynamically updated to reflect TEIRESIAS's corresponding growth in understanding.

III Contributions and Discussion:

TEIRESIAS upgrades the traditional "learning by being told" used in expert system construction and maintenance to "learning by knowing what to ask". This reflects a significant shift in responsibility for transfer of expertise from the domain expert to the expert system. Rather than passively accepting new, well-structured knowledge, TEIRESIAS actively converses with and aids the expert in formulating that knowledge. It not only facilitates learning by the expert system, but also learns itself as its models are automatically revised. This self-modification can be classified as "learning by experience" as a knowledge engineer.

TEIRESIAS's effectiveness as an intelligent assistant is based on its understanding of the knowledge and structure of expert systems. As such, it is envisioned as most suited for the later stages of construction and maintenance of mature expert systems. Furthermore, these expert systems must use the H. P. P. architecture. Within this restriction, TEIRESIAS can be tailored

to any domain with minimal alterations.

There are some weaknesses in the TEIRESIAS approach to transfer of expertise. The H. P. P. approach is based on the sufficiency of surface knowledge to build expert systems. As such, TEIRESIAS has a very shallow understanding of the semantics of new knowledge. It is incapable of checking the consistency of the new knowledge with the current version of the knowledge base. Also, TEIRESIAS's understanding of the expert system is based on its models of the current inference rules. This assumes correctness of the knowledge base and its qualification as a good basis for predicting future knowledge.

Papers: Meta-level Knowledge: Overview and Applications

Randall Davis and Bruce Buchanan
IJCAI, 1977.

Meta-knowledge and Cognition

Avron Barr
IJCAI, 1979.

I Problem:

In "learning by being told", the recipient of expert knowledge in the H. P. P. architecture accepts whatever is told on face value. Syntactic and semantic errors are passed along, unchallenged, by the learning interface. Specifically, the learning interface is blissfully unaware of what the format of the inference rules are, what the language of the domain is as used by the expert, and what the expert system already knows. The composite system (including the learning interface) has no introspection capacity. Consequently, the responsibility of system learning and maintenance rests solely on the design team. The concept of meta-level knowledge explicitly accessible to the learning element was developed and incorporated in TEIRESIAS to partially correct this learning deficiency.

II Approach:

The definition of object-level knowledge is that expert domain knowledge about things in the world. The definition of meta-level knowledge is that knowledge about object-level

knowledge which allows the learning element to use it, examine it, abstract it, reason about it, and direct its application. This concept of meta-level knowledge is essential to at least partially pass the responsibility of learning and system modification to the expert system itself.

TEIRESIAS employs several forms of meta-level knowledge. Each is an extended data type which describes a system data type, or knowledge chunk, explicitly to the system. The schemata correspond to the declarative world knowledge relevant to the expert domain. An example, within the context of an investment expert system, is the stock name schema. Schemata consist of a prototype framework, associated slotnames, and pointers to other associated schemata. The prototype and associated slotnames are used to prompt the expert for information when he introduces new declarative domain knowledge. The pointers are used to reference other data structures which might be affected by a new piece of this type of knowledge. In effect, the meta-knowledge in TEIRESIAS is very utility-oriented with an emphasis on structural knowledge. The schemata are static checklists for constructing known data types. In order to introduce a new data type, the design team would have to provide the corresponding schema.

While the schema is fairly static and provided by the design team, the rule model is dynamically maintained by the expert system. It is the analog of the schema, for procedural expert knowledge. The rule model represents a composite picture of a group of inference rules that conclude about a common domain

attribute. An example is the rule model which describes that group of rules which recommend investing in AT&T. Rule models consist of a prototype rule, common attributes, correlations among attributes, and pointers to other associated rule models. The decision of what constitutes a common attribute and when to correlate them is made by the system statistically based on the current membership of the group. The rule model is that meta-level knowledge which allows the system to "second-guess" and ask context-based questions to aid the expert in articulating his knowledge. It is a dynamic checklist.

A type of meta-level knowledge which serves a radically different function is the meta-rule. The meta-rule has the same format as object-level rules. It embodies the strategies, or preferences, that the domain expert has for directing the application of the object-level inference rules, based on the session context. Meta-rules are used to heuristically choose among candidate rule groups which could be used to extend the inference tree. Meta-rules are intended to constrain the exhaustive, depth-first search currently implemented in H. P. P. expert systems. An example is that blue-chip stocks should be considered first when the client is nearing retirement age.

III Contributions and Discussion:

The powerful idea of meta-knowledge is essential to provide the self-modification capability found in learning systems. It enables a form of introspection. The version employed by

TEIRESIAS allows it to participate in a mixed-initiative conversation to articulate a rule. TEIRESIAS has some idea of the syntax and semantics that the new rule should have, based on its understanding of the current state of the knowledge base. It's meta-knowledge is very ad hoc and mostly structural with utility emphasized. TEIRESIAS represents a start in researching meta-knowledge, but many problems remain. For example, the idea of strategic meta-knowledge is very interesting but not fully developed.

Paper: Model-Directed Learning of Production Rules

By: Bruce G. Buchanan and Tom M. Mitchell
Pattern Directed Inference Systems, 1978.

I Problem:

In systems where the knowledge representation method is the production rule a human expert is used to form the rule set for the system. Much work in the manipulation and interpretation of rules has been done, but can programs emulate the human experts ability to learn production rules by inducing these rules from empirical data?

Clearly if such programs are to be written they must have a strong world model to define interesting associations and to limit any hypothesizing process. In conjunction with such a model a learning strategy must be developed that balances general and specific concepts to form useful rules.

II Approach:

This work was done in the domain of mass spectrometry to form rules for an existing expert system called DENDRAL. Since this work represents knowledge about learning it is known as Meta-DENDRAL. Meta-DENDRAL is to examine data from the mass spectrometer and form rules describing the mass spectroscopic analysis of molecules. It is possible to define molecular structures in terms of atom types and bond characteristics such as the number of hydrogen neighbors or the number of double

bonds, thus forming a concept description language for the mass spectrometry process.

Since the mass spectrometer breaks a molecule into fragments which may regroup, and measures the relative abundance of various fragment masses, it is natural to attempt to form rules describing molecules in terms of fragments and their abundance. Thus a molecule and several fragment abundance measurements form a training instance. The operation of the mass spectrometer is described in a world model called the half-order theory that says such things as, "double bonds don't break" and other constraints on molecular fragmentation as well as constraints on atom migration.

The learning strategy is based on a plan-generate-test sequence. A set of related training instances is examined and using the world model groups of possible fragments are selected to fit the observed data. Molecules that might have originated these fragments are hypothesized in general terms. This process is a heuristic search from the most general molecule (i.e. two molecules bonded together) to more specific representations including atom type, number of bonds etc.

Heuristic analysis of rules is based on positive and negative evidence for the rule. Since these rules represent directed guesses at what a "good" rule might be they often fail to predict or falsely predict certain behavior. The analysis is performed by a critic function. Also during rule formation it is desired to form as complete and correct a set of rules as

possible thus there are attempts to improve rule performance by arbitrarily making some rules more general and some more specific. If a particular rule seems to explain the sum of what other rules explain it replaces the group of rules.

III Contributions and Discussion:

Through the use of a model to direct rule formation and with heuristic search along the general to specific concept spectrum, meaningful results have been generated to predict mass spectrometer behavior for several classes of molecules. Indeed Meta-DENDRAL performs like a well-educated spectroscopist and has contributed new knowledge about mass spectroscopy.

The use of a world model is at the center of most learning systems and is seen to play an important role as an aid to finding interesting concepts. The use of a general to specific ordering was later formalized into the version space approach which eliminates the use of heuristic searches as used in Meta-DENDRAL.

Paper: Version Spaces: An Approach to Concept Learning

By: Tom M. Mitchell
Ph.D. thesis, 1978.

I Problem:

Concept learning can be approached as a study of positive and negative examples, or instances, of the concept to be learned. The goal is to generalize the important features into a concept description. Thus the results of this learning by "induction" process would be the set of concept descriptions or "version space" consistent with the given set of training instances. However previous concept learning systems generated incomplete and possibly inconsistent sets of concept descriptions and thus lost some of the information in the training instances.

If concept learning is viewed as a search of possible concepts to fit the training instances then the previous methods may be viewed as: depth-first heuristic search, that is try to guess the concept based on an evaluation of the "best" concept description, or breadth-first heuristic search which uses the notion of a "general" to "specific" ordering. Version spaces are an elaboration and a formalization of this ordering as a mathematically precise partial ordering and thus eliminate some difficulties with the previous methods. Such difficulties are determining when concepts are consistent with all training instances, when concepts are ambiguous, when there are inconsistencies in the training set and what training instances best define the concept.

II Approach:

The version space approach is a method for defining the concept solution space by computing boundary sets of maximally specific and maximally general concept descriptions. These concept descriptions must be in a finite or at most countably infinite concept description language. This language is domain dependent.

The version space concept learning algorithm is called the "candidate elimination algorithm" and takes as input training instances, and outputs updated concept description boundary sets. This algorithm is based on four functions and a pattern matching predicate. The pattern matcher identifies concept descriptions that fit training instances. Two of the functions generate the set of allowable maximally specific and maximally general concept descriptions for a training instance and the remaining two functions update the boundary sets with these concept descriptions. The processing is primarily dependent on whether the training instance is a negative or positive example of the concept.

Because of its mathematically formal definition this approach may be proven correct, that is the boundary set representation of the version space is a complete representation and the candidate elimination algorithm generates the proper boundary sets for a given training instance. This mathematical base allows analysis of many of the characteristics of this approach.

III Contributions and Discussion:

With the use of the candidate elimination algorithm and version spaces the total set of concept descriptions consistent with the training instances is generated, thus all information is fully used. This may cause problems if the training instances are inconsistent, but some techniques for dealing with this problem are a part of the version space approach. The computational load is lessened in comparison to previous methods in that past training instances need not be reviewed to validate updated concepts. The version space is a provably correct and complete bounding process instead of a heuristic guess. As a result of the bounding the order of presentation of training instances is irrelevant and indications of future valuable training examples are available.

One area of concern is the size of the boundary sets. The application of the version space approach to the Meta-DENDRAL program in which it replaced a complex heuristic search process has shown as good or better results with large but manageable boundary sets.

The Heuristic Programming Project and Learning

I Brief Analysis of the H. P. P. Approach:

It should be noted that the H. P. P. architecture supports the only performance programs currently competitive with human experts in certain fields. MYCIN, for example, is an expert system in diagnosis of blood infections and meningitis infections and the recommendation of drug treatment. It is the prototypical H. P. P. expert system. In the system validation phase, MYCIN's performance was rated by a panel of domain experts. In 90% of the test cases analyzed, a majority of these judges indicated that MYCIN's decisions were the-same-as or as-good-as their own.

Some problems exist. For example, the 90% figure associated with MYCIN contains, in large part, the typical, routine cases experts handle almost off-handedly. The tougher cases requiring deeper thought and more intricate analysis lie in the 10% not matched. This is a serious and inherent limitation of the approach. H. P. P. researchers have committed themselves to surface knowledge sufficiency. They are trying to model a human expert at a shallow level. The consequence is short-term success with an asymptotic performance limit established by a combination of the ability and articulation of the most productive expert. This introduces the second limitation associated with the approach, the process of knowledge extraction. Since there is no attempt to formulate and understand the deeper knowledge in the domain, knowledge extraction is an art rather than a systematic,

engineering task.

Even assuming that the H. P. P. approach is the correct one, there is a limitation associated with the architecture. The performance curve, as a function of the number of rules in the knowledge base, indicates that MYCIN's surface knowledge capabilities cannot be easily extended. It takes more and more rules to produce the same amount of performance improvement.

II Analysis of the H. P. P. Research on Learning:

H. P. P. work in learning originated as support for their prime research goal, developing expert system design concepts within the H. P. P. approach. As such, the H. P. P. perspective is one of the most applications-oriented in the state of the art of learning systems.

The programming tool, EMYCIN, the AI tool librarian, AGE, and the intelligent assistant, TEIRESIAS, collectively exhibit a trend. This trend is towards more and more automation of the design team's programming responsibilities. These systems collectively raise the knowledge base construction process from employing a rigid Lisp format to a restricted, but free-form, subset of natural language. They handle the bookkeeping and detail work involved. They also provide the domain-independent explanation capabilities which can be used to examine system reasoning in a debugging context. In all cases, the expert system under construction learns by being told. In AGE, the

domain expert will also learn under AGE's intelligent guidance thru library documentation of available building tools. In TEIRESIAS, the intelligent assistant will also learn by maintaining up-to-date models which mirror the current state of the knowledge base. Within the limited range established by their development-oriented goals, the H. P. P. efforts in learning have succeeded.

The machinery of learning exhibited by meta-DENDRAL is more ambitious than that of TEIRESIAS, AGE, or EMYCIN. The goal is to learn by induction. This ability would not only remove the knowledge engineer from the picture, but also would assume most of the domain expert's responsibilities in knowledge articulation. Meta-DENDRAL is more powerful. This is due in part to the domain-dependent learning structure employed by meta-DENDRAL as opposed to the architecture-dependent learning structure used in the others. The added power is gained at the expense of the generality of the learning structure.

Some problems exist. H. P. P. efforts have been concentrated in the knowledge base construction phase of expert system construction. The most difficult phase of this expert system construction, knowledge extraction from the domain expert, has been left relatively untouched. One aspect of knowledge extraction, that it is an art, makes it a much more difficult domain for learning research. Other aspects, that it is a teaching process and that it is the stopgap of expert system construction, makes it almost imperative to formalize the

process, preferably thru imposing a matching learning structure. Meta-DENDRAL is one attempt to automate knowledge extraction thru induction. However, its research success is very much based on the domain characteristics and cannot easily be generalized or adapted to other expert domains.

Other problems surface as inherent limitations of the approach. Since learning is at the surface knowledge level, the learning systems which monitor the knowledge base construction process cannot check for knowledge consistency. The learning system cannot verify whether the new rule to be added will cause the expert system to draw contradictory conclusions, or some resident rules to never again be used. A related problem is knowledge completeness. The learning system cannot use introspection of deeper concepts to identify holes in its one-level knowledge fabric. The expert system either has the pertinent rules or the learning system is unaware that such knowledge even exists. This has ramifications in knowledge extraction. The determination of areas within the expert domain which should be examined for new rules is currently still an entirely human process. Lastly, knowledge redundancy can clutter up the knowledge base. The learning systems have no rule comparison capability to determine that several rules overlap or are actually encompassed in a more general version.

The short-term success of the H. P. P. architecture as a basis for expert systems has severely biased their learning research. Meta-DENDRAL research was initiated in response to

heuristic DENDRAL needs in 1972. TEIRESIAS research addressed MYCIN's requirements and produced a Ph.D. thesis in 1976. AGE and EMYCIN research progress was reported in 1979. This trend indicates that the performance nature of H. P. P. learning research is unlikely to change in the near future.

III Relevance to Air Traffic Control:

By necessity, the comments, pointers, and questions concerning the relevance of the H. P. P. learning systems research to the air traffic control (ATC) problem will take the form of educated conjecture. Generally speaking, the level of expertise reflected in the ATC domain is comparable with that of the expert domains considered by H. P. P.' Also, the performance achieved by H. P. P. expert systems is at a very high level, and therefore an appropriate goal.

Some more specific conclusions can be reached. The power exhibited by meta-DENDRAL due to its domain-dependent learning structure makes a strong case for a similar domain-dependent structure in ATC. The meta-level of knowledge made explicit in TEIRESIAS and meta-DENDRAL for introspection is certainly necessary. It's syntactic aspects is the basis for self-modification by any system. Furthermore, its semantic aspects could facilitate deeper, more critical learning thru model-directed induction of ATC strategies from case studies. The context-based questioning capability that TEIRESIAS displays could also prove invaluable for seeking and challenging guidance

in this induction process.

The numerous questions raised by this analysis of H. P. P. learning research include those that address the classification of types of meta-knowledge. What other types are there beyond that which encodes structure knowledge? What subdivisions of meta-knowledge exist for the ATC domain within the one used by Davis to include all levels of strategies? What is gained by a deeper understanding and organization of the ATC domain knowledge? How can this knowledge's consistency and completeness be insured? These are among the issues that must be resolved on the way to an expert-level performance learning system in the ATC domain.

Section III

M. I. T.

Artificial Intelligence Lab

Overview of M. I. T. Papers

Most of the M. I. T. works have been related to basic research which, unlike Stanford's Heuristic Programming Projects, are not intended for immediate applications. Consequently, there is less continuity among the subjects being discussed.

Both Winston's and Sussman's Ph.D. theses are about experiments in the BLOCKS world, which is a domain that is semantically simpler than those of the real world. Although research conducted in the BLOCKS world has been questioned for its extensibilities, both papers did discuss some important issues that any real-world task using similar approaches may have to face.

The important conclusion from Winston's ARCH work is that the use of "near-miss" examples is a powerful technique in that the instructor will be able to convey precisely the essential ideas to the underdeveloped model. Also Winston discusses the complications of having negative training instances that are not so "near" to the existing model. Since single-difference near-miss examples are not so common in the real-world we find from his experience that a carefully devised training sequence, starting with simple but definite near-miss examples, can avoid confusion when learning from more complicated examples.

Sussman's thesis deals with skill acquisition (or learning procedures). Although only the performance (the speed of problem-solving), not the "competence" of the HACKER is improved through the learning practice, Sussman suggested an interesting way to modify and to generalize the existing procedures. Since problem-solving in the ATC environment is basically dealing with the learning of procedures, his experience is particularly helpful to our work in ATC learning.

Goldstein's "Annotated Production System" paradigm is an improvement of the traditional production system which has been widely utilized as a tool to model the behavior of domain experts. The commentary formalism added to the standard production-rule format makes the annotated production system more likely to cope with complicated real-world situations. Furthermore, we share Goldstein's concern about the efficiency of a real-time system. The proposed "heuristic compiler" to improve the run-time efficiency of the knowledge base can be classified as "learning by self-reorganization".

Finally, Winston's recent contribution is about creating and justifying "transfer frames" as a method of learning. The computer student is given a pair of related concepts, called "simile", by the human instructor. The computer program, the student, is to figure out the ideas that the instructor is trying to get across. Winston proposes the idea of "transfer frame" as a filter of information flow between the source and the destination concepts. If learning in the ATC environment is to

be achieved through human instruction, the idea of "learning through simile" is quite applicable in that the computer can be taught to solve an ATC problem by refering to similar problems with typical solutions.

Paper: Learning Structural Descriptions from Examples

By: Patrick Winston
Ph.D. Thesis, 1970.

I Problem:

To describe an abstract concept is always a challenging task for those people who are interested in artificial intelligence as well as psychology. In the case of artificial intelligence, the goal is not only to understand the concept, but also to build up a computer model for such a concept so that the computer program can make use of the model to perform some intelligent tasks. Working within the domain of three-dimensional structures of the BLOCKS world, Winston proposes a way to build up the description of a visual scene, and to establish the model of a visual concept through a carefully devised training sequence. The significance of this work extends beyond the BLOCKS world as it addresses several important issues related to computer vision and computer learning.

II Approach:

The initial step of the concept learning process is to build up the structural description for a visual scene. To begin with, Winston performs a preliminary scene analysis by applying the vision programs written by H. N. Mahabala and A. Guzman. Mahabala's program classifies and labels the vertices and regions of a scene, and the result is feeding to Guzman's program which,

in turn, groups the vertices and regions together into labeled objects. The resulted visual information, in the form of vertices, regions, and objects, is taken as input to Winston's description-building programs. Relationships and properties of the objects, such as ABOVE, IN-FRONT-OF, SUPPORT-OF, LEFT-OF, RIGHT-OF, LARGE, SMALL, STANDING, LYING, are discovered by their corresponding procedural detectors. After building up the description of the local relationship, grouped objects are recognized by a 2-step process of conjecture- criticism and revision. Up to this point, a hierarchical description of the scene is established.

Concept-learning is achieved by introducing a sequence of positive (examples) and negative (counter-examples) instances of the to-be-learned concept, each instance being pre-processed to establish their structural descriptions. The model of the concept is gradually built up by comparing current instance with the already-existent model. Special emphasis is on the "differences" between these two structural descriptions. In the case of "examples" (positive instances), each difference implies a possible generalization of relationship in the model. For negative instances, the differences are those necessary conditions (MUST-BEs) which characterize the model. Winston proposes a powerful notion, called "near-miss", which is a negative instance in a training sequence quite like the concept to be learned but differs from the concept in only a small number of significant points. Those small differences permit the machine to localize some part of its current opinion about a

concept for improvement. Through the use of such "near-miss" instances, the teacher can convey particular ideas quite clearly.

III Contributions and Discussion:

Through the study of structural learning process in the domain of BLOCKS world, Winston demonstrates the powerfulness of using "near-miss" instances to convey a definite idea to the under-developed model. In a supervised learning environment with human as teacher, it is the responsibility of the human instructor to carefully devise the "near-miss" training instance. This implies that the teacher must understand the internal structure of the model so that he can effectively convey the essential ideas.

In the domain of air traffic control, if learning is to be achieved under human supervision, Winston's experience is of particular value. The results point out that if effective learning is to be achieved through a sequence of near-miss instances, the trainer must not only be an ATC expert, but also be a computer expert who is familiar with the internal model of the intelligent ATC program.

Paper: Learning by Creating and Justifying Transfer Frames

By: Patrick Winston
Artificial Intelligence:
An M. I. T. Perspective, 1980.

I Problem:

The transfer frame method of learning falls under the category of "learning by studying samples". It is a descendant of a similar sample studying approach known as near-misses. The student is active in the learning process and shares the burden with the teacher.

II Approach:

The learning system (student) initially possesses knowledge represented as frames. The frame is able to represent descriptions of almost anything, objects, actions, etc. For example, a cube may have slot-value pairs such as color-blue and material-wood. The teacher supplies an instruction in the form of a similarity, <destination> is like <source>, known as a simile. It is desired to add knowledge about the <destination> from that of the <source>, which something is known about. This is accomplished with the creation of a transfer frame, which passes or "filters" the information meant to be transferred from the <source> to the <destination>. The information represents some commonality. Transfer frames are created in two steps, hypothesis followed by filtering. During hypothesis, a list of

possible or "candidate" slots is made from analysis of the <source> and its fellow class members (relatives). Filtering isolates the correct slots whose values are to be transferred from analysis of the <destination> and its relatives.

The many criteria for choosing candidate slots are applied in order. Slots with exceptional (extreme) values is the most typical candidate slot. Next, slots know to be globally important are selected. Finally, outstanding slots of the <source> in terms of existence and value with respect to its relatives are chosen. The candidate slots are grouped into transfer frames according to properties.

Filtering is also performed by an ordered application of rules. Transfer frames are chosen which have slots typical among the <destination> and its relatives. Because of connection between successive similes, previous transfer frames are also used.

III Contributions and Discussion:

Learning with transfer frames facilitates the acquisition of knowledge with simple similarity instructions. With a sequence of similes, it is possible to rapidly build a frame for something know little about. The main limitation is dealing with descriptive knowledge only in the form of frames. The ATC problem can be broken down into a hierarchy of plans, general control strategy at the top, followed by flight plans, and

real-time aircraft maneuvers. For each level, it is possible to attach descriptions to its members and expand its knowledge base. For example, flight paths can be broken down into descriptions relating to the crew, controller, passengers, and aircraft. Possible slot values are performance (crew,controller) and fuel efficiency (aircraft). There are frames for particular flights, and frames for new flights can be generated with transfer frames.

Paper: A Computational Model of Skill Acquisition

By: Gerald J. Sussman
Ph.D. Thesis, 1973.

I Problem:

An important subject in the domain of artificial intelligence is to understand the principles behind the intelligent problem-solving activities that humans perform daily. A skill is a set of answer procedures to achieve a definite goal. HACKER is a computer problem-solving system to model the process underlying the development of such skills. Along with a library of the developed skill, HACKER has the ability to develop a new skill with its general problem-solving, debugging, and learning capabilities. Because of the learning capability, the performance of HACKER improves with practice.

II Approach:

Just as a human problem-solver, when attacking a problem, HACKER first tries to classify the problem into a subclass for which a solution method has already been developed. If an old skill is available, it will be invoked to solve the problem. However, if none of the developed solutions is applicable, HACKER will make up a new solution using some general problem-solving techniques, the so called "bag of tricks", applied to his knowledge of the domain. In doing so, HACKER utilizes critics to check for inapplicable situations which were summarized, or

learned, from its previous problem-solving experience. If it turns out that the new solution proposed by the general strategies fails to solve the problem, HACKER enters the debugging phase to localize the cause of such failure. It reviews the history of current problem-solving process and pinpoints the causes of failure. The "bug" is then classified into known type by comparing with the prototype bugs in the library. A remedial action is thus suggested to modify the problem-solving process. If this process results in a successful plan to solve the given problem, the new solution, or "skill", is stored in the answer library along with its problem pattern. In addition, the bug triggers a learning process by adding critics into the library so that, if later encountering the same type of problem, the problem-solver can be expected to avoid making these mistakes again.

III Contributions and Discussion:

HACKER is the first problem-solving system attempting to use the planning "bugs" instead of avoiding them. The idea underlying HACKER is to learn from an incomplete plan, bug-ridden as it can be, by analyzing the causes of its failure. Learning comes in both positive and negative sides, i.e., either to "remember" the solution method to a new problem, or to modify the strategy of the problem-solver to avoid making the same mistake again. The former being not so attractive because, in the long run, the developed skills are going to flood the answer library

so that it is inefficient to search for possible developed solutions. This is in direct conflict with the principle of intelligent problem-solving, which is to encode the methodology of problem-solving into the program so that the solution can be developed on the spot rather than to memorize each solution for every problem. The latter learning strategy is more interesting. It suggests to abstract the bugs and readjust the behavior of the problem-solver by posting exceptional conditions to its general problem-solving techniques. Such learning strategy has potential application in the domain of en-route air traffic control. As the air traffic controllers are guided by some general strategies to solve the encounted air traffic problems, a journeyman controller must be able to recognize the exceptional situations, an ability that he "learned" from his many years of experience. Such skill coincides with what HACKER is addressing. Since HACKER is designed to solve problems within the domain of BLOCKS world which is quite limiting in its semantic depth, it is premature to say that the ideas elucidated by HACKER are directly applicable to the domain of air traffic control. Nevertheless, the study about the relationship between "problem-solving" and "learning" by Sussman is a helpful experience toward our work of solving the controller's problems.

Paper: Annotated Production Systems:
A Model for Skill Acquisition

By: Ira Goldstein and Eric Grimson
IJCAI, 1977.

I Problem:

A procedural model for modeling of skill acquisition in student pilots is proposed. The skill in acquisition is attitude instrument flying. For this application, a production system model is inadequate. The model consists of standard production rules with additional formal commentaries (meta-knowledge).

Production systems have demonstrated desirable characteristics in their generality, modularity, and simple control. However, in skill acquisition they have severe limitations. For example, there are inefficiencies in dealing with context, a lack of direction for debugging, and no self-knowledge for learning. Learning ability and performance are severely hampered.

The task involved is attitude instrument flying (AIF). Typical subgoals are maneuvers such as steady climbs, turns, descents, and level flight. From observations of instrument readings (e.g. airspeed), the student pilot applies control to the aircraft. Here, the mapping process from measurement to control is more sophisticated than other domains. Characteristics are higher order effects, interrelated control, and context-sensitivity. The real-time nature of the task also demands fast response times and consequently higher performance.

II Approach:

3.1 Production Systems

Production systems (PS) are composed of production rules of the form <pattern> invokes <action>. Goals are achieved by recognition of patterns and execution of actions. This works for a sequence of independent recognize-act pairs. Because AIF is context-sensitive, independence does not hold and poor performance results. Some rules are more likely to be applied and others have exceptions. Dependence between actions require communications between productions, which has limited support in PS. Consideration of future consequences of actions is also not possible in PS. The result is that the sequential paradigm in achieving a goal may fail and more complex procedures such as coordinate routines are needed. The ordering of rules may add additional conditions in the application of a rule, increasing response time.

3.2 Annotated Production Systems

The need for second-order knowledge results in an annotated production system model (APS). Annotations to production rules are in the form of caveats, rationales, plans, and control information. Caveats provide information regarding exceptions to rules, planning, and interrelationships among rules. Control annotations resolved the situation where several rules may be used to achieve a goal. Rules are ordered as being primary,

checking, and backup. Rationales are placed globally and locally to aid performance debugging via explanation. Annotations are used to verify appropriateness and success in execution of a rule and also in failure recovery.

Learning is accomplished by modifying production rules with help from annotations. Control specialists are generated form global and local control annotations to direct rule application. Caveat specialists are also formed to check and look for exceptions. Learning occurs from generalizing rules and analogy of rule patterns and actions.

III Contributions and Discussion:

PS were originally designed for psychological modeling of individuals. Realistic domains such as AIF require higher order knowledge in the form of annotations. It also facilitates an environment for "learning by analogy" and "learning by generalization". The task of an air traffic controller is similar to that of AIF; both are real-time situations and direct aircraft trajectories. However, effects are higher-order due to interaction between other aircraft and their controllers.

Summary of M. I. T. Works

In the M. I. T. works reviewed, various forms of learning have been discussed. ARCH is a case of "learning from examples". It achieves the construction of a computer model through a sequence of carefully devised examples from the instructor. In HACKER, the instructor merely assigns some "exercises" without further comment, and the system gradually improves its problem-solving ability through practice. This is a case of "learning by self-discovery", with a minimum assistance from the instructor. However, as discussed before, the "improvement" is mostly on the "speed" of problem-solving rather than on the "competence". The APS (Annotated Production System) proposed the idea of using the "heuristic compiler" to reorganize its own knowledge base, and hopefully resulting in a more efficient knowledge base for problem-solving. It can be considered as a case of "learning by self-reorganization". Again, it is the efficiency of problem-solving that is of main concern, not the competence.

Winston's "Transfer Frame" system is another case of "learning from examples". The instructor attempts to convey to the computer student a new concept by refering to a related known-concept. It is the responsibility of the computer student to figure out what the key concepts such analogies are trying to point out. The difficulties arise mostly from the impreciseness of the communication between the teacher and the student, which

is more likely a subject of linguistic research.

The M. I. T. works mostly deal with innovative ideas without clear intended applications. Although theoretical in nature, their experience with various types of learning can be helpful to our research on the ATC applications.

Section IV

Carnegie-Mellon University

Overview of C. M. U. Papers

The purpose of research at Carnegie-Mellon University is to understand intelligent systems and the nature of human intelligence. This can be viewed as a combined social science and artificial intelligence approach with roots in the Logic Theorist, a 1956 Newell and Simon publication that describes a computer program that imitates human problem solving with heuristic search. During this period, learning was a popular research topic in artificial intelligence. Most researchers held the view that it was easier to induce a system to organize itself from scratch, by exposing it to an appropriate set of training instances, than it was to provide it with the knowledge it would need for expert performance. These efforts somewhat paralleled the theories of Hebb and other psychologists that proposed that the basis of learning was largely the self-organization of neural nets.

During the past ten years there has been a resurgence of research in learning. In a recent article, Simon states "The principle mechanism of intelligence that we have observed (in people or computers) operating in problem environments is heuristic search". Combined with research into heuristic search algorithms is the current paradigm of programming a model of expert behavior to include an initial database of declarative knowledge, i.e., rules of the domain. Characteristic of the new research into learning is the use of production systems.

Production systems are sets of conditional rules and are used extensively at Carnegie-Mellon University to model human behavior in several domains. Examples include Lenat's AM (acquires new mathematical concepts which in turn helps to acquire others), Langley's BACON (induces scientific laws from data using recursive procedures), Neve's research (learns skills by analysis of worked out textbook examples), and Anderson's ACT (simulates language acquisition and learning of geometry proof skills).

Paper: Automated Theory Formation in Mathematics

By: Douglas B. Lenat
IJCAI, 1977.

I Problem:

The formulation of theories in any scientific discipline requires both (1) the discovery of interesting relationships among known concepts, and (2) the invention of new concepts. The first of these activities has been demonstrated in the meta-DENDRAL system wherein regularities in chemical mass spectrometry data are discovered and encoded into compact rules, and the latter activity by Winston in a system which learns new concepts about structures composed of blocks (such as arch or tower) by being shown examples and non-examples. Lenat proposed combining these two types of activity and implemented his ideas in a system called AM which discovers patterns and creates new concepts in elementary mathematics.

II Approach:

AM consists of two distinct bodies of knowledge: a network of concepts and a collection of heuristic rules. The heuristics look for patterns within and among the concepts, and those which are particularly interesting become newly created concepts; hence, the network of concepts grows larger.

Initially, AM contained about 100 concepts from elementary set theory. These were organized in a hierarchy with the most general concepts at the top. Among the more general concepts are "object" and "activity". Under "object" are the concepts of "set", "empty-set", "ordered-set", etc., while under "activity" are "predicate", "operation", and many particular operations like "set-intersection".

Each concept is composed of many facets such as the name of the concept, its definition, and an algorithm for its execution if it happens to be an operation. Many facets are simply pointers to other concepts. In fact, these pointers are what ties the concepts together into a network. Among these facets are generalizations, special cases, and examples of the concept. Other facets contain more complicated relationships involving several other concepts as in analogies and conjectures.

Although the knowledge about mathematics is encoded in the network of concepts, the real knowledge is embodied in the heuristics. Each heuristic has an "if" part which must be satisfied before the heuristic will be used, and a "then" part which can have one of three kinds of effects: (1) fill in the contents of one facet of a concept, (2) create a new concept, or (3) suggest a new task. In fact, most tasks are of the form "fill in a particular facet of a particular concept" because that facet is currently empty. If, for example, the task is to fill in the "examples" facet, then the heuristic which uses the "algorithm" facet to generate examples is called into operation.

However, other heuristics may also be pertinent (i.e., their "if" parts are satisfied). Thus, if only one example is found, that example could become a new concept; if the set of examples found is similar to the set of examples of another concept, then conjecture that that concept is identical to the one being worked on; if very many examples are found, suggest finding examples of a special case of the concept; and so forth.

In order to keep AM from wasting too much time on fruitless endeavors, all the tasks to be performed are ordered in a queue according to their priority. A task is given higher priority (1) if it was suggested on more than one occasion, (2) if it is related to a particularly interesting concept, and (3) if it was suggested very recently. The interestingness of a concept is just another facet of a concept and is therefore operated on by heuristics. For example, a heuristic which creates a new concept is likely to increase the interestingness of all generalizations of that concept. The last of the above conditions causes AM to focus its attention on a given line of discovery, much as humans do.

III Contributions and Discussion:

The discoveries made by AM are indeed impressive (for a machine). Starting only with elementary set theoretic concepts, AM discovered counting, numbers, addition, multiplication, division, primes, unique factorization, and much more. Contributing to this success was the use of symbolic reasons

attached to each task. Thus, the number of different reasons why a task was suggested could readily be evaluated in order to determine what priority it should be given.

However, unlike the network of concepts, the set of heuristic rules could not grow. Since the original set of rules were mainly applicable to set theory, their power rapidly diminished as the system delved deeper into number theory. Unique factorization is about as far as AM was able to go. It became clear that in future systems of this sort, it would be necessary to have rules which could add new heuristics tailored to deal with the newly created concepts.

Paper: Rediscovering Chemistry with BACON.4

By: Pat Langley, Gary Bradshaw, and Herbert A. Simon
Workshop on Current Developments
in Machine Learning, 1980.

I Problem:

One clear example of "learning by discovery" is that of abstracting natural laws from empirical data. Quite frequently, the laws involve many variables which makes it difficult to find the formula that relates them by trial and error alone. To aid in the discovery of these laws, scientists employ a number of heuristic rules which reduce considerably the number of formulas that must be tried. An attempt to find these heuristics and to implement them in a computer program was the apparent purpose of a system called BACON at Carnegie-Mellon University.

II Approach:

The usual technique in designing an experiment is to hold all the variables constant except for a few (usually two) for which a numerical relationship is being sought. This is also the technique employed by BACON. If a predictable relationship is found between the remaining variables, i.e., their product is a constant, or one varies linearly with respect to the other, then the parameters of that relationship are hypothesized to be new physical properties and thus become new variables which can be calculated from and used as data in other experiments.

To illustrate this process, consider an experiment consisting of a sealed bottle of some gas also containing a thermometer and barometer. A linear relationship would be noted between the temperature T and pressure P. The slope and intercept of the line obtained by plotting T versus P are considered to be constants (perhaps depending on the experimental conditions). In this case, the intercept is the temperature corresponding to absolute zero, while the slope is a constant which depends on the volume and contents of the bottle. The latter dependency would be discovered by performing similar experiments with different bottles and different quantities of gas.

If it was not immediately known that the volume of the bottle was a relevant factor, then a label corresponding to the bottle and its contents (and any other conditions that might affect the results) would be treated as a nominal variable as opposed to a numerical variable. It would be noted that the slope of T versus P varies depending on the value of the nominal variable while the value of the intercept does not. Thus, the value of absolute zero would be hypothesized to be a constant, while some as yet undiscovered property of the bottles and gases would be hypothesized to explain the differing values of the slope. In fact, the value of the slope would be defined to be the value of this new property. The next step would be to find a relationship between this new property and, say, the volume of the bottle (keeping the amount of gas constant). This would result in postulating yet another property which would be found

to depend on the amount of gas used in the experiment. After discovering that relationship, the system will have finally arrived at the ideal gas law.

III Contributions and Discussion:

Earlier versions of BACON were able to discover the correct formulas for Snell's law of refraction, conservation of momentum, and gravitational attraction using the same set of heuristic rules. Application of the system to chemistry, however, required the addition of another heuristic rule, namely, that of looking for common submultiples among a set of data points. After that, BACON was able to retrace most of the more basic discoveries made by early chemists.

The whole point of this system seems to be how many discoveries can be made with a fixed set of heuristic rules. However, there is a great deal more to scientific discovery than the discovery of formulas to explain experimental data. One must create models of the underlying phenomena in a given experiment, and given the model, more detailed experiments can be designed to refine the model, and so forth. Eventually, a model may have to be rejected, e.g., Rutherfords's model of the atom as a planetary system, but without it, we may never have discovered the current model (quantum mechanical). Thus the next step for automated scientific discovery should at least consider model formation.

Papers: A General Learning Theory and Its Application to the Acquisition of Proof Skills in Geometry

John R. Anderson
Workshop on Current Developments
in Machine Intelligence, 1980.

A Learning System and Its Psychological Implications

John R. Anderson and Paul Kline
IJCAI, 1979.

I Problem:

The authors contend that there are three types of learning required for a student to learn skills such as planning and generating geometric proofs. In the context of how a student searches for a proof tree, these include the acquisition of operators in procedural form, learning to represent problems in a way that operators can more easily be applied, and tuning operators so that they will apply more appropriately. Operators are production rules and theories of learning are examined with a computer simulation titled ACT.

II Approach:

ACT is a production based computer simulation based on a semantic network. Knowledge is divided into two categories: declarative and procedural. Declarative knowledge is represented in a propositional network similar to Quillian's semantic network. Procedural knowledge is represented as a set of productions. ACT's control structure is an iteration through

successive cycles, where each cycle consists of a production selection phase followed by an execution phase. On each cycle an APPLYLIST is computed which is a probabilistically defined subset of productions whose conditions have all their active constants in the data base. The probability that a production will be placed on the APPLYLIST depends on the strength of that production relative to the sum of the strengths of the other productions.

ACT can learn by adding propositions to its data base and by adding productions. It can also learn by modifying the strengths of its productions. Productions can be added by encoding of instructions or by the restructuring of productions in response to experience.

When students first learn a definition or theorem it is not immediately converted to procedural knowledge. Part of this is due to confusion on how best to use the new knowledge. A good student will learn to plan proofs using forward and backward reasoning combined with pattern recognition based on experience. The process of translating declarative knowledge into productions is called procedural compilation.

Finding the proof tree for any given geometric proof problem requires an efficient search procedure. Four theories on how to make the forward and backward reasoning more effective are based on analogy, generalization, discrimination, and composition. Analogy uses the solutions that worked in past similar solutions. Generalization forms a new rule based on what two problems have

in common. Discrimination is complementary to generalization in that restrictions are placed on a rule's applicability. Composition will create a new production that accomplishes the effect of a sequence of previous productions.

III Contributions and Discussion:

The articles are a broad overview of learning research by an integrated research team of artificial intelligence researchers and cognitive psychologists. Many of the concepts described in the papers are not new to the field of knowledge acquisition or knowledge representation. Their contribution is an approach to the study of learning which represents a shift from concentration on empirical study to computer simulation based on artificial intelligence concepts.

Paper: Learning to Use Analogies

By: John McDermott
IJCAI, 1979.

I Problem:

The author states that in order for a system to learn how to do new tasks it must be capable of assimilation and accommodation. Assimilation enables a system to relate unfamiliar situations to situations that it knows about. The unfamiliar situation is temporarily transformed into a familiar situation. Accommodation enables the system to make such transformations permanent. A computer simulation program, ANA, is used to examine these learning theories.

II Approach:

If a system is to learn it must be capable of assimilation and accomodation. Initially, ANA has limited knowledge about how to function in a simple environment. ANA is a production system. The productions are rules with a condition part and an action part (i.e., IF ....., THEN .....). Sets of productions that solve a unique problem are called methods. ANA learns by analogy. When confronted with an unfamiliar situation, it maps the description for which it has a method into the description of the unfamiliar task. As it uses the method it executes the actions dictated by the mapping. If the modified method is used successfully, it is stored as a method. If unsuccessful, ANA

attempts to patch the method. This is done either by trying to incorporate a more specific method or asking a task master for help.

ANA's learning strategy is to find a method that is adequate for a related task and assume it will be adequate for the unfamiliar task. This can easily lead to errors. ANA has several built in productions to prevent these error conditions. One production is set up initially to test the validity of any goal or sub-goal. In cases where the mapping is under specified, ANA either tries to incorporate another method or asks for help. Over specified mappings require the intervention of the task master.

The simple environment used in this paper is the toy domain of a paint shop. There are 18 rooms (L01 - L18), 19 objects (6 different types) of various weight, color, configuration (i.e., stacked), and state (clean or dirty). There are five operators (spray, carry, push, cart, and scan) and several rather contrived constraints (e.g, no more than four boxes to a room). The knowledge base initially consists of six methods. One method describes how to paint tables red that are in L32. The other methods are for transporting objects. For instance, one of these methods tells how to move boxes that are in L25 to an unspecified location. The two unfamiliar tasks solved by ANA are (1) "Paint the blue chair in L21 red and move it to L35" and (2) "Wash the thing in L12".

III Contributions and Discussion:

ANA's learning is at best adequate in that it learns only enough to get by. This is a simple, but effective learning technique in toy domains where goals are described with certainty. This may not be transferrable to more abstract domains such as playing chess or directing air traffic. The learning is very dependent on the methods stored in the knowledge base and a task master (i.e. teacher) may be required. It is significant that ANA asks questions when it thinks it needs help rather than being told what to do. A serious weakness appears to be that ANA has no knowledge of how to select an appropriate method. This will manifest itself when the number of methods becomes large or an unfamiliar situation is significantly different than anything seen before.

A short comparison should be made with the HACKER program of Sussman. ANA and HACKER attempt to find new solutions to problems based on stored solutions. The chief difference is in their method building strategy. Hacker examines code and rewrite procedures. ANA attempts to incorporate more specific methods or asks for help (patches).

Papers: Patterns of Induction and Associated Knowledge Acquisition Algorithms

Frederick Hayes-Roth
C. M. U. Technical Report, May 1976.

Theory Driven Learning: Proofs and Refutations as a Basis for Concept Discovery

Frederick Hayes-Roth
Workshop on Current Development in Machine Learning, 1980.

I Problem:

In the former paper, the problem considered is very similar to those of traditional pattern recognition. Given examples of characters from a sample of text, learn what features are needed identify a character, or given a much longer sample of text, by observation of recurring patterns, infer various word categories such as adjectives and learn the rules of grammar. In the latter paper, the author describes learning that evolves from the consideration of proofs and refutations.

II Approach:

In the pattern recognition type of problems, a data structure called a parameterized structural representation (PSR) is proposed as a very general method for describing the essential characteristics needed to identify any kind of object or category. A method called interference matching is described which takes several similar PSRs and produces an abstraction of them, i.e., a more general PSR which includes only their common

characteristics. It can therefore be said that these more abstract categories have been learned from examples.

In the second paper it is postulated that a central kind of learning in mathematics occurs when counterexamples arise that refute theorems. Possibly the most productive type of counterexample is one that satisfies the conditions of the theorem but fails to satisfy its conclusions. The bulk of the paper is concerned with the descriptions of heuristics used in proof rectification. Briefly, these include: (1) introducing an additional necessary condition to preclude applications of the refuted theory in situations characteristic of the counterexample, and (2) plan modification that generates alternative actions which avoid the counterexample.

III Contributions and Discussion:

Although the earlier paper suggested a technique for "learning by example", no mention was made of using counterexamples. The scheme described could presumably abstract the common characteristics of an upper case "A" from a number of examples; however, it was incapable of modifying its abstraction given that characters like "B" or "H" were not examples. This serious deficiency was certainly corrected in the latter paper, although the problem domain was quite different. Instead of operating on a database of descriptions of patterns, it deals with procedural entities, e.g., sets of production rules which can be viewed as theorems. The techniques described, however,

are not a substantial improvement over the work of McDermott; although, as the work is still in progress, new results may be forthcoming.

Analysis and Summary of C. M. U. Work

Most of the work in artificial intelligence at C. M. U. is very much oriented toward production system architectures. This has certainly been evident in the work specifically related to machine learning. Although other centers of research (e.g., Stanford Univ.) have chosen production systems because they are especially appropriate to certain task domains, at C. M. U. the motivation for their use is more psychological. Their hypothesis is that the human mind is functionally organized in a system of production rules, and their research is one means of testing this hypothesis. If a production rule system on a computer exhibits human-like behavior, then the hypothesis is strengthened. The papers by Langley and Anderson strongly conform to this view of the work at C. M. U. The other papers also conform to this view, but to a lesser extent.

It is curious to note that two of the papers, Lenat's and Langley's, were oriented toward autonomous "learning by discovery", i.e., learning without any assistance from an instructor. This is considered to be the most advanced classification of learning, and these two papers are foremost among the few attempts at this kind of learning. The successes of these two learning systems showed very clearly the limitations of the current approaches. From them, it has been learned that the set of heuristic rules which augments the database of domain knowledge (e.g., mathematics or physical laws) must also be

augmented. This implies the need for heuristic rules which themselves create new heuristic rules, and of course it would not be surprising if we found that heuristic rules one level above those were required, and so on ad infinitum.

It should be pointed out that both Lenat's and Langley's learning systems worked within well-structured domains. The real world, on the other hand, is much less well-structured, not to mention that what is "true" in the real world varies with time. The approaches toward learning by Anderson and in the more recent work by Hayes-Roth are much more applicable to the real world. Their approaches allow production rules to be incrementally modified, either because of a changing environment or because the rules have not yet been exposed to all possible circumstances.

Another important aspect of learning systems in practical problem areas is that of rule arbitration. When a given situation satisfies the conditions (if-part) of more than one rule, there ought to be some additional criteria for deciding which of these rules should be tried first. The most general of these criteria is the "context", i.e., the present state of the world. Clearly, this is something which can and should be learned (from past experience). Such a scheme was employed in Anderson's system wherein the type of geometric proof to be solved would cause the selection of those rules most likely to lead to the solution most quickly. There, the context was simply the type of geometric proof under consideration, and from past experience, the system was able to recall which rules led to

success in similar situations. For all its simplicity, this idea is very powerful. However, when applied to more complicated domains (e.g., air traffic control), the context may be much more difficult to characterize.

Section V

Other Work in Machine Learning

Overview of Other Work in Machine Learning

Research in machine learning has experienced a very rapid increase in interest recently within the study of artificial intelligence. This interest has manifested itself within two basic categories. One can be called Engineering AI in the sense that the internal operation of the learning system is secondary in importance to the performance of the learning system. In other words, the criteria for judging machine intelligence is based on its ability to exhibit its competence through performance. The machine is regarded as a black box. The other basic category can be called Cognitive AI in that the internal operation of the learning system is at least as important as its performance. In other words, the criteria for judging machine intelligence is based on the resemblance of the machine's learning structure to the human's learning structure, in addition to the machine's performance. It is important to note that the learning structure in humans is not yet completely understood, leading to various approaches which claim to model the same structure. But in any event, approaches within the two categories share a common testing criterion, namely implementation on the computer.

From the previous sections of this survey, one can see that each approach surveyed has characteristics of both Engineering AI and Cognitive AI. However, a sense of the relative importance attached to each category can be inferred and used to categorize

the work of the Stanford Heuristic Programming Project as more Engineering AI, and the work of Carnegie-Mellon and M. I. T. as more Cognitive AI. These institutions show a continuity in history and effort towards research in machine learning which is unique within the field. Other institutions surveyed do not share the volume and connectedness of the work offered by these institutions. Nonetheless, they also have many interesting works to consider. This section will survey a sampling of these.

One system, REDHOT, considers the dialogue of supervised learning from the point of view of the student, but with a new twist. Whereas other supervised learning systems are preprogrammed with an implicit understanding of a unique teaching methodology to be used, REDHOT considers that there might be several such methodologies. Each would be used in appropriate situations depending on the message that the learning sequence is intended to impart. The intelligent student must be able to recognize the teaching strategy after seeing the first few examples of the sequence and use that recognition to facilitate learning. REDHOT can be classified as a Cognitive AI system.

The Semantic Net Matcher (SNM) is a tool used within the PROSPECTOR expert system by the design team to maintain and update the knowledge base. As a learning tool rather than a learning system, it has limited capabilities. However, the SNM does address the problem of knowledge base consistency after the addition of new chunks of knowledge, an issue of great importance which is still unsolved. SNM can be classified as an Engineering

AI system.

KLAUS is intended to be a learning system. The key concept offered by KLAUS is that of a seed system which defines a preprogrammed level of understanding. This seed system is also preprogrammed with a set of meta-level judgemental knowledge which defines its inquisitiveness as new knowledge is introduced. With the addition of a fixed set of syntactic/semantic rules to drive its limited natural language interface, KLAUS attempts to simultaneously learn both new domain knowledge and the linguistic constructions used to communicate that knowledge. KLAUS can be classified as Engineering AI.

The analogical reasoning system of Chen and Findler represents one implementation of "learning by analogy" which is highly restrictive of the conditions under which it will perform. This paper is included because it shares some insight into analogical learning and reasoning, a strategy which is potentially the most versatile. Analogy can be considered as the bridge between different expert domains as opposed to much of the current work in learning within one domain. Chen and Findler's work can be classified as Engineering AI.

Papers: Modelling Student Acquisition of
Problem-solving Skills

Robert Smith
Rutgers University
NCAI, 1980.

Modelling Student Acquisition of
Problem-solving Skills: The Student's
Interpretation of the Teaching Environment

Robert Smith
Rutgers University
Workshop on Current Developments
in Machine Learning, 1980.

I Problem:

Previous works in "learning thru examples" have always concentrated on the heuristics students use to interpret the content of examples. This may be thru analogy, induction, near-miss, simile, and so on. In other words, the lessons presented to the student comprised the totality of stimulus for knowledge acquisition. The characteristics of the teacher were ignored. The scheme for the particular sequencing of examples was implicit and fixed. The teacher was restricted to one teaching format. The role he played in the learning dialogue was rigid and underemphasized. But there are good teachers and there are bad teachers. This seems to indicate the possible existence of diverse teaching strategies. Therefore, the intelligent student, to get the most out of a lesson, must also be aware of the underlying teaching strategy being used. The teacher must have the flexibility of choosing the structure of his lessons. REDHOT is an attempt to introduce and utilize this meta-level

knowledge, which adds a dimension to the teacher's role in student learning. This variant on "learning by examples" is called "learning by teaching".

II Approach:

The problem which REDHOT addresses is that of skill acquisition in constructing proofs in elementary logic. The model of good teaching it attempts to simulate is taken from a computer-aided instruction (CAI) course in this domain offered at Stanford University. The format of each lesson clearly delineates explanatory text, examples, exercises, and hints.

REDHOT begins with certain basic knowledge and capabilities at the primitive level. The initial set of primitive operators is complete in that there is sufficient knowledge to solve any problem. Basically, the set includes the usual rules of inference. The goals of REDHOT, what it will learn, is to build macro operators from the initial set and to generate application heuristics which indicate when these macro operators are to be used. These application heuristics are similar in intent to the meta-rules envisioned in TEIRESIAS.

REDHOT currently recognizes two learning strategies, "focus and elaborate" and "focus and frustrate". An example of their application in learning is the use of "focus and elaborate" to extend the primitive modus ponens operator (i.e. deducing "B" given both "A" and "A implies B") to a macro modus ponens

operator. This macro is generalized to handle an arbitrary number of iterations of the primitive modus ponens operator. The ordered sequence of examples leads off with several exercises which involve a straight one-step application of the primitive operator. This might be called the focus step, which highlights the modus ponens operator as the one which will be elaborated on. The next exercises would require a two-step application, and then a three-step application, and so on. From this sequencing of exercises, REDHOT would infer that the intent of the teacher is to teach the generalized modus ponens operator. REDHOT then generates the appropriate macro operator.

What is envisioned is a library of teaching strategies with which the student system can recognize the intent of the teacher. This library would add another dimension to the learning process by making these teaching strategies explicit. Currently, learning system interfaces between teacher and student are restricted to one implicit, hard-wired teaching strategy. Emphasis of learning research has been on the inference mechanisms which act on the content of the lessons. REDHOT is an attempt to recognize and use the meta-level intent of the lesson structure.

III Contributions and Discussion:

REDHOT is an important step in the direction of upgrading the importance of the teacher's role in the knowledge acquisition process. The characteristics which distinguish a good teacher

from a bad teacher include the orderliness of a lesson. The intelligent student must ask himself, "why did he say that just now" and interpret the contents of the lesson accordingly. However, the clear classification of the teaching strategies used by good teachers is a difficult task, for which, the identification of the "focus and elaborate" strategy must be considered a small, first step.

Paper: Using a Matcher to Make an Expert Consultation System Behave Intelligently

By: Rene Reboh
SRI International
NCAI, 1980.

I Problem:

One of the major considerations in designing an expert system which improves its skills thru knowledge acquisition is that of insuring knowledge base consistency. How does a new chunk of knowledge fit into the model of the expert system knowledge. If that new chunk of knowledge is not consistent with the extant body, subsequent use of the knowledge base could very well yield false advice. Thus, the interaction of each newly-acquired piece of knowledge with the current state of the knowledge base is an important consideration for the expert system design team. The Semantic Network Matcher (SNM) partially addresses this task for the PROSPECTOR expert system. The SNM checks the interaction of each new knowledge chunk with the knowledge base and flags certain types of inconsistencies for the design team to correct.

II Approach:

PROSPECTOR is an expert system which serves as a consultant for mineral exploration in much the same manner as MYCIN does for blood disease diagnosis. Its knowledge base is organized as an explicit network of statements connected by rules or logical

constructs - a partitioned semantic network representation. The SNM currently supports three features which aid in keeping the knowledge base error-free and consistent in form and content as it is expanded by the design team.

Since such a network serves as the basis for judgemental reasoning, various numerical values, such as probabilities of certainty, are maintained to direct the consultation. The SNM is partially responsible for maintaining probabilistic consistency. Suppose S2 is the most recently entered statement in the semantic net, and it is a restriction of S1 (i.e. S1 implies S2), where S1 is a statement already in the knowledge base. Since S2 is a restriction, its associated probability can never exceed that of S1. If it does, the SNM flags the situation for correction. This is not always sufficient. Suppose S1 and S2 now appear in two production rules in PROSPECTOR: E1 --> NOT(S1) and E2 --> S2. Then the inconsistent situation in which probability $P(S1|E1) < P(S2)$ might occur. In other words, when the rules were introduced by the design team, they overlooked the fact that E1 should be equally unfavorable for S2, and E2 should be equally favorable for S1. The SNM recognizes such a potential problem which it again flags for the design team.

New statements which are entered by the design team must match the granularity of the current knowledge base. Also, for a knowledge base which employs various representation schemes, the design team must decide which one to use for a particular chunk of knowledge. Which representation, and at what granularity, is

finally chosen depends on what other statements are in the knowledge base and how they are related to the new statement. Since the SNM can be used to assist in analyzing how statements are related, it plays a role in this process which accompanies knowledge acquisition.

Finally, in constructing and maintaining a knowledge base, the design team cannot be expected to remember the entire contents of that knowledge base. The SNM can be used here as a search tool. Given a partial description of the statement desired, the SNM can find the appropriate one thru content-driven search.

III Contributions and Discussion:

The SNM addresses a difficult and crucial problem in knowledge base construction and maintenance. This issue of knowledge base consistency has not been solved. The approach offered here is piecemeal and not intended to be complete. The use of probability in judgemental reasoning opens the gates to probabilistic inconsistencies at every level of the consultation process. The types of inconsistency checked here are straightforward and at a very shallow level of rule interaction. In fact, the SNM is a tool which, while versatile, is very basic.

Paper: An Approach to Acquiring and Applying Knowledge

By: Norman Haas and Gary Hendrix
SRI International
NCAI, 1980.

I Problem:

The systems which form the state of the art in expert domain problem-solving, such as MYCIN, DENDRAL, and PROSPECTOR, have been carefully assembled by hand. This construction scenario has required two types of expert knowledge, expert knowledge within the domain and expert knowledge of the system's knowledge representation structure. TEIRESIAS is an attempt to establish an intelligent assistant which is expert in the latter type of knowledge. EMYCIN and AGE also fall into this category. All corresponded to various perturbations of the H. P. P. architecture as the knowledge representation structure. All are independent of the domain knowledge being formalized. All act as interfaces for the domain expert to try to fit his knowledge into a prefabricated representation scheme. KLAUS also addresses this problem: how to enable a computer system to acquire new knowledge in new domains from tutors who are experts in their respective fields, but who have little or no training in computer science. KLAUS is another example of "learning by knowing what to ask".

II Approach:

KLAUS is in effect an interface, expert in the target knowledge representation, for the domain expert. It simultaneously learns new concepts and the new linguistic constructions used by the tutor to express these concepts. It operates in mixed-initiative interactive mode by asking for detail and in general, applying its meta-level judgement criteria to ask for these clarifications in an intelligent manner. These and other primitive concepts are called seed concepts which, along with seed vocabulary, form the initial learning knowledge base of KLAUS. The initial learning structure, deductive algorithms and a fixed set of syntactic/semantic rules, is also preprogrammed.

As an example of the application of the approach to learning suggested by KLAUS, a system which learns to become expert in Naval logistics was created. This seed system is called NANOKLAUS. NANOKLAUS's target performance system has a knowledge structure and application goals which are different from that of TEIRESIAS's target performance system. Instead of rules which are used to drive inferential reasoning, the NANOKLAUS knowledge base will become a large aggregation of facts which are used to drive a question/answer system. These facts are encoded as well-formed formulas in the first order predicate calculus knowledge representation. NANOKLAUS is the learning system for an intelligent information management system. However, the KLAUS approach of an initial seed knowledge base, learning structure,

and meta-level inquisitiveness is domain independent.

The initial NANOKLAUS system knows about certain seed concepts such as PHYSICAL OBJECT, PERSON, and MEASURE, and certain seed vocabulary such as "unit", "kind", and "plural", which are used frequently in stating definitions of new concepts and new words. NANOKLAUS also is preprogrammed with certain meta-level judgement knowledge such as asking for the relationship between two concepts, "feet" and "meters", which are of the same type, MEASURE. It uses five principles of knowledge organization to integrate new knowledge: (1) there are things, (2) there are subclasses of things (leading to a heritance hierarchy), (3) there are relations among things, (4) there are subclasses of relations (leading to a heritance hierarchy), (5) some of the relations are functions. From this preprogrammed knowledge level, NANOKLAUS learns such things as the physical hierarchy among ships, responsibilities associated with each rank, and so on.

III Contributions and Discussion:

Like TEIRESIAS, KLAUS is an interface between the expert and the target system. It is expert in the target knowledge representation and assumes that the expert is not. Unlike TEIRESIAS which is envisioned for helping to update and maintain a mature system, KLAUS is intended for the initial expert system construction phase. However, the preprogramming of the seed system is a very difficult task unless the programmer is expert

in the target domain. For information management, which is more well-understood and orderly, Klaus may be quite useful. However, for more heuristic domains such as medicine, what constitutes the seed knowledge base in procedural knowledge and what constitutes inquisitiveness are very difficult questions to answer.

Paper: Analogical Reasoning in Problem-solving

By: David Chen and Nicholas Findler
SUNY at Buffalo
IJCAI, 1977.

I Problem:

Analogical learning and reasoning is based on the idea that two situations which are similar in some ways are likely to be similar in other ways too. Similar situations call for similar actions. The application of analogy can be within one domain, solving a new problem using the solution of a similar problem as a starting point. It can be between domains, recognizing problem similarities between two different domain problems on a common level of feature extraction. As a learning strategy, analogy has the potential to bridge expert domains allowing a sharing of problem-solving expertise.

II Approach:

The analogical learning and reasoning system presented in this paper is based on many working hypotheses. First, a key step in analogy is recognizing and extracting the similarity features that are "important". This step is assumed to be already done and is not addressed in this paper. Secondly, the assumption is made that the number of problems that share a feature indicates the importance of that feature. Of course, these problems must have similar solutions in order to be considered in weighting the relative worth of the common feature.

Lastly, the more features shared by two problems, the more similar their solutions will be.

The knowledge base is composed of solution segments indexed by the important problem features. This knowledge base can be described in terms of two models which also drive the learning structure. The contributive model actually stores the <features> --> <solution fragment> knowledge chunks, along with a history of the frequency of successful usage of that chunk. This past "experience" is a heuristic guide to the order of selecting among candidate fragments to use in generating an actual solution segment. The hierarchial model is an ordered tree in which features are arranged by relevance. This importance weighting of features also serves as heuristic guide in determining the order of selecting among candidate fragments for an actual soultion segment. The search for a solution segment is exhaustive until a satisfactory one is found, if that segment already exists. In the sense that these models are statistical and therefore dynamic, this system is said to learn.

The analogical learning and reasoning structure is envisioned to work within a problem-solver in the following way. A separate module handles the problem-solving logic, the decomposition of the problem into subproblems. This decomposition is made with the recommendations of the analogical module. They both communicate with a data base that encompasses the domain-specific knowledge.

III Contributions and Discussion:

This paper imparts a partial understanding of the learning strategy of analogy. It indicates the complexity of creating a systematic learning structure based on this strategy. It also has severe shortcomings. First of all, feature "importance" is a very crucial part of analogy. Criteria for judging when a feature might be important, and when it might not, seem to form the larger part of the analogy process, but is not touched on in this paper. Secondly, no procedure for automatic knowledge acquisition of solution segments is provided. In fact, the only learning that is done is in rearranging the order in which a fixed library of solution segments is to be searched. Lastly, this implementation of analogical reasoning is applicable only to problems whose solution segments are independent. No segment instantiation to accomodate a slight change in requirements is provided. Analogy can be a very powerful learning structure, and much more work needs to be done.

Analysis of Work in Machine Learning

From the papers presented in this section, as well as those in the previous sections, it should be clear that the problem of machine learning is the target of renewed interest. From the initial knowledge base construction to its maintainance and enhancement, learning aspects exist in all phases of the expert system evolution. This survey gives us a good understanding of what the state of the art is, and a feel for what needs to be done with respect to the air traffic control problem.

Many unsolved problems have been analyzed, including the key one of automatic maintainance of knowledge base consistency. Some work has been attempted in solving this consistency problem by regarding it an engineering effort to solve certain known problem aspects within the selected architecture and the context of the performance system. Other works have attacked the problem from another direction, regarding consistency as a separate main issue. Nonetheless, this problem must be solved to facilitate expert domain problem-solver construction.

Several aproaches to constructing the expert system have been offered, such as a system incorporating a seed learning knowledge base and a seed learning structure. With this initial structure, the seed system intelligently asks for and about new knowledge in a mixed-initiative dialogue. However, the difficulty of defining just exactly what is a seed level of an

expert domain is hard to gauge. This is especially true of domains which are characterized as heuristic and procedural in nature, such as medicine.

Various learning strategies has been partially clarified, including inductive learning and analogical learning. Analogy is a powerful strategy which can be used to transform a solution in one expert domain to an analogous solution in a different domain, thereby providing a bridge for sharing of expertise. It needs to be more fully investigated.

Within the generic learning scenario, many enhancement issues have been raised, such as the meta-level knowledge of intent displayed by the instructor in teaching. What is the teacher trying to say? Obviously, an intelligent student must be able to recognize that intent to extract the message behind the lesson sequence.

From this survey, it should be clear that some interesting results have been obtained in the area of machine learning. It should also be clear that much work remains to be done, in understanding learning and in applying that understanding to the air traffic control domain.

Section VI

Bibliography

Aikins, J. S., Representation of Control Knowledge in Expert Systems, NCAI, August 18-20, 1980, pp. 121-123.

Aikins, J. S., The Use of Models in a Rule-Based Consultation System, IJCAI, Vol. 1, August 22-25, 1977, pp. 788.

Aikins, J. S., Prototypes and Production Rules: An Approach to Knowledge Representations for Hypothesis Formation, IJCAI, Vol. 2, August 20-23, 1979.

Akaama, K. and A. Ichikawa, A Basic Model for Learning Systems, IJCAI, Vol. 2, August 20-23, 1979.

Amarel, S., Report of AIM (AI in Medicine) Workshop, IJCAI, Vol. 1, August 22-25, 1977, pp. 993.

Amarel, S., J. S. Brown, B. G. Buchanan, P. Hart, C. A. Kulikowsi, W. A. Martin and H. Pople, Applications of Artificial Intelligence, IJCAI, Vol. 1, August 22-25, 1977, pp. 994-1006.

Anderson, J. R., A General Learning Theory and Its Application to the Acquisition of Proof Skills in Geometry, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Anderson, J. and P. Kline, A Learning System and Its Psychological Implications, IJCAI, Vol. 2, August 20-23, 1979.

Anzai, Y., N. Ishibashi, Y. Mitsuya and S. Ura, Knowledge-Based Problem Solving By a Labelled Production System, IJCAI, Vol. 2, August 20-23, 1979.

Aubin, R., Strategies for Mechanizing Structural Induction, IJCAI, Vol. 1, August 22-25, 1977, pp. 363-369.

Baisley, A., Tech II, IJCAI, Vol. 1, September 3-8, 1975, pp. 165.

Ball, E. and P. Hayes, Representation of Task-Specific Knowledge in a Gracefully Interacting User Interface, NCAI, August 18-20, 1980, pp. 116-120.

Blazer, R., L. Erman, P. London, and C. Williams, HEARSAY-III: A Domain-Independent Framework for Expert Systems, NCAI, August 18-20, 1980, pp. 108-110.

Banerji, R. B., Requirements on and Problems in Description Languages and Induction, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Banerji, R. B. and G. W. Ernst, A Comparison of Three Problems-Solving Methods, IJCAI, Vol. 1, August 22-25, 1977, pp. 442-499.

Barr, A., Meta-Knowledge and Cognition, IJCAI, Vol. 2, August 20-23, 1979.

Barstow, D. E., The Roles of Knowledge and Deduction in Program Synthesis, IJCAI, Vol. 2, August 20-23, 1979.

Barstow, D., Knowledge-Based System for Automatic Program Construction, IJCAI, Vol. 1, August 22-25, 1977, pp. 382-388.

Barstow, D. R., Knowledge Engineering in Nuclear Physics, IJCAI, Vol. 2, August 20-23, 1979.

Bauer, M., A Basis for the Acquisition of Procedures from Protocols, IJCAI, Vol. 1, September 3-8, 1975, pp. 226-231.

Bennett, J. S. and R. S. Engelmore, SACON: A Knowledge-Based Consultation for Structural Analysis, IJCAI, Vol. 2, August 20-23, 1979.

Berliner, H. J., Making Judgments, NCAI, August 18-20, 1980, pp. 134-137.

Berliner, H., On the Construction of Evaluation Functions for Large Domains, IJCAI, Vol. 2, August 20-23, 1979.

Berliner, H. J., A Chronology of Computer Chess and Its Literature, Artificial Intelligence, Vol. 10, No. 2, April 1978, pp. 201-214.

Berwick, R. B., Learning Structural Descriptions of Grammer Rules From Examples, IJCAI, Vol. 2, August 20-23, 1979.

Bibel, W., On Syntax-Directed, Semantics-Supported Program Synthesis, IJCAI, Vol. 2, August 20-23, 1979.

Bobrow, D. G., G. G. Hendrix, W. A. Martin, J. McCarthy, A. Newell, R. C. Smith, and N. S. Sridharan, Knowledge Representation, IJCAI, Vol. 1, August 22-25, 1977, pp. 983-992.

Bobrow, D. B., R. M. Kapla, M. Kay, D. A. Norman, H. Thompson and T. Winograd, GUS, A Frame-Driven Dialog System, Artificial Intelligence, Vol. 8, 1977, pp. 155-174.

Bobrow, R. J. and B. L. Webber, Knowledge Representation for Syntatic/Semantic Processing, NCAI, August 18-20, 1980, pp. 316-323.

Bobrow, D. G., T. Winograd and KRL Research Group, Experience with KRL-0: One Cycle of a Knowledge Representation Language, IJCAI, Vol. 1, August 22-25, 1977, pp. 213-222.

Bonzon, P., Learning of Abstractions from Structural Descriptions of Pictures, IJCAI, Vol. 2, August 20-23, 1979.

Briabrin, V. M., The Dialogue Information Logical System, Machine Intelligence 9, 1978, pp. 427-444.

Brooks, R. and J. Heiser, Controlling Question Asking in a Medical Expert System, IJCAI, Vol. 2, August 20-23, 1979.

Brown, D. J. H., Concept Learning by Feature Value Interval Abstraction, Proc. Workshop Pattern-Directed Inference Systems, SIGART Newsletter 63, 1977, 55-60.

Brown, F. M. and S.-A. Tarnlund, Inductive Reasoning in Mathematics, IJCAI, Vol. 1, August 22-25, 1977, pp. 844-850.

Brown, J. S., Steps Toward Automatic Theory Formation, Proc. 3rd Int. Joint Conf. Artificial Intelligence, Stanford, CA, 1973, 121-129.

Buchanan, B. G., Issues of Representation in Conveying the Scope and Limitations of Intelligent Assistant Programs, Machine Intelligence 9, 1978, pp. 407-426.

Buchanan, B. G., Scientific Theory Formation by Computer, Proc. NATO Advanced Study Inst. Computer Oriented Learning Processes, Noordhoff, Leydon, 1976.

Buchanan, B. G., E. A. Feigenbaum, and N. S. Sridharan, Heuristic Theory Formation: Data Interprettion and Rule Formation, Machine Intelligence, Vol. 7, (Meltzer, B., and Michie, D., eds.), Edinburgh Univ. Press, Edinburgh, 1972.

Buchanan, B. G. and T. Mitchell, Model Directed Learning by Production Rules, Pattern-Directed Inference Systems.

Buchanan, B. G., D. H. Smith, W. C. White, R. J. Gritter, E. A. Feigenbaum, J. Lederberg and C. Djerassi, Automatic Rule Formation in Mass Spectrometry by Means of the Meta-DENDRAL Program, J. Am. Chem. Soc. 98, 1976.

Buchanan, B. G., G. Sutherland and E. A. Feigenbaum, Heuristic Dendral: A Program for Generating Explanatory Hypotheses in Organic Chemistry, Machine Intelligence, Vol. 4 (Meltzer, B., and Michie, D., eds.), America Elsevier, New York, 1969, 209-254.

Buchanan, J. R. and R. D. Fennell, An Intelligent Information System for Criminal Case Management in the Federal Couts, IJCAI, Vol. 1, August 22-25, 1977, pp. 901-902.

Bundy, A., Will it Reach the Top? Prediction in the Mechanics World, Artificial Intelligence, Vol. 10, No. 2, April 1978, pp. 129-146.

Burstall, R. M. and J. A. Goguen, Putting Theories Together to Make Specifications, IJCAI, Vol. 1, August 22-25, 1977, pp. 1045-1058.

Carbonell, J. G., Interactive Concept Acquisition on Hierarchical Memory Structures, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Cercone, N. and L. Schubert, Toward a State Based Conceptual Representation, IJCAI, Vol. 1, September 3-8, 1975, pp. 83-90.

Chandrasekaran, B., F. Gomez, S. Mittal and J. Smith, An Approach to Medical Diagnosis Based on Conceptual Structures, IJCAI, Vol. 2, August 20-23, 1979.

Charniak, E., A Partial Taxonomy of Knowledge About Action, IJCAI, Vol. 1, September 3-8, 1975, pp. 91-98.

Chavchanidze, V., Heuristico-Conceptual Programming, IJCAI, Vol. 1, August 22-25, 1977, pp. 919.

Chen, D. T. and N. V. Findler, Analogical Reasoning in Problem Solving, IJCAI, Vol. 1, August 22-25, 1977, pp. 345.

Chien, R. and S. Weissman, Planning and Execution in Incompletely Specified Environments, IJCAI, Vol. 1, September 3-8, 1975, pp. 169-174.

Chiu, W. Y., Structure Comparison and Semantic Interpretation of Differences, NCAI, August 18-20, 1980, pp. 259-262.

Clancey, W. J., Dialogue Management for Rule-Based Tutorials, IJCAI, Vol. 2, August 20-23, 1979.

Cohen, B. L., A Powerful and Efficient Structural Pattern Recognition System, Artificial Intelligence, Vol. 9, 1977, pp. 223-256.

Davis, R., Interactive Transfer of Expertise: Acqusition of New Inference Rules, Artificial Intelligence, Vol. 12, No. 2, August 1979, pp. 121-158.

Davis, R., Knowledge Acquisition in Rule-Based Systems: Knowledge About Representation as a Basis for System Construction and Maintenance, Pattern-Directed Inference Systems.

Davis, R., Interactive Transfer of Expertise: Acquisition of New Inference Rules, IJCAI, Vol. 1, August 22-25, 1977, pp. 321-328.

Davis, R. B., Representing Knowledge About Mathematics for Computer-Aidedd Teaching, Part I - Educational Applications of Conceptualizations from Artificial Intelligence, Machine Intelligence 8, 1977, pp. 363-386.

Davis, R., B. Buchanan, and E. Shortliffe, Production Rules as a Representation for a Knowledge-Based Consultation Program, Artificial Intelligence, Vol. 8, 1977, pp. 15-46.

Davis, R., Application of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases, Memo AIM-283, Artificial Intelligence Laboratory, Stanford Univ., Stanford, CA, 1976.

Davis, R. and B. G. Buchanan, Meta-Level Knowledge Overview and Applications, IJCAI, Vol. 1, August 22-25, 1977, pp. 920-927.

Davis, P. R. and R. T. Chien, Using and Re-Using Partial Plans, IJCAI, Vol. 1, August 22-25, 1977, pp. 494.

Davis, R. and J. King, An Overview of Production Systems, Machine Intelligence 8, 1977, pp. 300-334.

de Mori, R. and L. Saitta, Scheduling of Processes in a Speech Understanding System Based on Approximate Reasoning, IJCAI, Vol. 2, August 20-23, 1979.

de Kleer, J., The Origin and Resolution of Ambiguities in Causual Arguments, IJCAI, Vol. 2, August 20-23, 1979.

de Kleer, J., Multiple Representations of Knowledge in a Mechanics Problem-Solver, IJCAI, Vol. 1, August 22-25, 1977, pp. 229-304.

DeJong, G., Prediction and Substantiation: Two Processes that Comprise Understanding, IJCAI, Vol. 2, August 20-23, 1979.

Dershowitz, N., Automatic Program Annotation, IJCAI, Vol. 1, August 22-25, 1977, pp. 378.

Dietterich, T. G., Multiple-Model Induction in Eleusis, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Dietterich, T. G. and R. S. Michalski, Learning and Generalization of Characteristic Descriptions: Evaluation Criteria and Comparative Review of Selected Methods, IJCAI, Vol. 2, August 20-23, 1979.

Doyle, J., A Glimpse of Truth Maintenance, IJCAI, Vol. 2, August 20-23, 1979.

Egan, D. E. and J. G. Greene, Theory of Rule Induction, Knowledge and Cognition (Gregg, L. W., ed.), Wiley, New York, 1974, 43-103.

Elshout, J. J. and B. J. Wielinga, A Computational Approach to the Study of Human Skill Acquisition, IJCAI, Vol. 2, August 20-23, 1979.

Engelman, C., M. Bischoff, and C. Berg, KNOBS: An Experimental Knowledge Based Tactical Air Mission Planning System and a Rule Based Aircraft Identification Simulation Facility, IJCAI, Vol. 2, August 20-23, 1979.

Engelman, C., E. A. Scarl and C. H. Berg, Interactive Frame Instantiation, NCAI, August 18-20, 1980, pp. 184-186.

Engelmore, R. S. and A. Terry, Structure and Function of the Crysalis System, IJCAI, Vol. 2, August 20-23, 1979.

Fagan, L. M., J. C. Kunz, E. A. Feigenbaum and J. J. Osborn, Representation of Dynamic Clinical Knowledge: Measurement Interpretation in the Intensive Care Unit, IJCAI, Vol. 2, August 20-23, 1979.

Farley, A. M., The Coordination of Multiple Goal Satisfaction, IJCAI, Vol. 1, August 22-25, 1977, pp. 495.

Faught, W. S., Modelling Intentional Behavior Generation, IJCAI, Vol. 2, August 20-23, 1979.

Feigenbaum, E. A., The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering, IJCAI, Vol. 1, August 22-25, 1977, pp. 1014-1029.

Feldman, J. A., Report on AlxPL (SIGART-SIGPLAN) Conference, IJCAI, Vol. 1, August 22-25, 1977.

Fickas, S. and R. Brooks, Recognition in a Program Understanding System, IJCAI, Vol. 2, August 20-23, 1979.

Fikes, R. E., P. E. Hart and N. J. Nilsson, Learning and Executing Generalized Robot Plans, Artificial Intelligence 3, 1972, 251-288.

Forgy, C. and J. McDermott, OPS, A Domain-Independent Production System Language, IJCAI, Vol. 1, August 22-25, 1977, pp. 933-939.

Fox, M. S., Reasoning with Incomplete Knowledge in a Resource-Limited Environment: Integrating Reasoning and Knowledge Acquisition, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Fox, M. S., On Inheritance in Knowledge Representation, IJCAI, Vol. 2, August 20-23, 1979.

Fox, M. S. and R. Reddy, Knowledge Guided Learning of Structural Descriptions, IJCAI, Vol. 1, August 22-25, 1977, pp. 318.

Fredkin, E., D. Michie, J. McCarthy and A. Newell, Invited Panel on Future Directions of AI, IJCAI, Vol. 1, August 22-25, 1977, pp. 1093.

Friedland, P., Knowledge-Based Experiment Design in Molecular Genetics, IJCAI, Vol. 2, August 20-23, 1979.

Friedman, L., Trouble-Shooting by Plausible Inference, NCAI, August 18-10, 1980, pp. 292-294.

Gaschnig, J., An Application of the Prospector System to DOE's National Uranium Resource Evaluation, NCAI, August 18-20, 1980, pp. 295-297.

Gaschnig, J., Preliminary Performance Analysis of the Prosppctor Consultant System for Mineral Exploration, IJCAI, Vol. 2, August 20-23, 1979.

Gaschnig, J., A Problem Similarity Approach to Devising Heuristics: First Results, IJCAI, Vol. 2, August 20-23, 1979.

Genesereth, M. R., An Automated Consulant for MACSYMA, IJCAI, Vol. 1, August 22-25, 1977, pp. 789.

Genesereth, M. R., The Role of Plans in Automated Consultation, IJCAI, Vol. 2, August 20-23, 1979.

Georgeff, M., A Framework for Control in Production Systems, IJCAI, Vol. 2, August 20-23, 1979.

Gladun, V. P. and Z. L. Rabinovich, Formation of the World Model in Artificial Intelligence Systems, Machine Intelligence 9, 1978, pp. 299-312.

Goldstein, I., Bargaining Between Goals, IJCAI, Vol. 1, September 3-8, 1975, pp. 175-180.

Goldstein, I. P. and E. Grimson, Annotated Production Systems: A Model For Skill Acquisition, IJCAI, Vol. 1, August 22-25, 1977, pp. 311-317.

Good, I. J., Summing Up of the Discussion of Inductive Inference, Machine Intelligence, 8, 1977, pp. 205-208.

Granger, R. H., Jr., When Expectation Fails: Towards a Self-Correcting Inference System, NCAI, August 18-20, 1980, pp. 301-305.

Green, C. and D. Barstow, Some Rule for the Automatic Synthesis of Programs, IJCAI, Vol. 1, September 3-8, 1975, pp. 232-239.

Green, C. C., R. P. Gabriel, E. Kant, B. I. Kedzierski, B. P. McCune, J. Phillips, S. T. Tappel, and S. J. Westfold, Results in Knowledge-Based Program Synthesis, IJCAI, Vol. 2, August 20-23, 1979.

Greiner, R. and D. B. Lenat, A Representation Language Language, NCAI, August 18-20, 1980, pp. 165-169.

Grinberg, M. R., A Knowledge Based Design System for Digital Electronics, NCAI, August 18-20, 1980, pp. 283-285.

Grosz, B. J., The Representation and Use of Focus in a System for Understanding Dialogs, IJCAI, Vol. 1, August 22-25, 1977, pp. 67-76.

Haas, N. and G. G. Hendrix, An Approach to Acquiring and Applying Knowledge, NCAI, August 18-20, 1980, pp. 235-239.

Haas, N. and G. Hendrix, An Approach to Acquiring and Applying Knowledge, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Hardy, S., Synthesis of LISP Functions from Exaples, IJCAI, Vol. 1, September 3-8, 1975, pp. 240-245.

Hart, P., Progress on a Computer Based Consultant, IJCAI, Vol. 1, September 3-8, 1975, pp. 831-841.

Havens, W. S., A Procedural Model of Recognition, IJCAI, Vol. 1, August 22-25, 1977, pp. 264.

Hawkinson, L., The Representation of Concepts in OWL, IJCAI, Vol. 1, September 3-8, 1975, pp. 107-114.

Hayes, P. J., On Semantic Nets, Frames and Associations, IJCAI, Vol. 1, August 22-25, 1977, pp. 99-107.

Hayes, P., A Representation for Robot Plans, IJCAI, Vol. 1, September 3-8, 1975, pp. 181-188.

Hayes, P. and R. Reddy, Graceful Interaction in Man-Machine Communication, IJCAI, Vol. 2, August 20-23, 1979.

Hayes-Roth, B. and F. Hayes-Roth, Concept Learning and the Recognition and Classification of Exemplars, J. Verbal Learning Verbal Behav. 16, 1977, 321-338.

Hayes-Roth, B., F. Hayes-Roth, S. Rosenschein, and S. Cammarata, Modeling Planning as an Incremental Opportunistic Process, IJCAI, Vol. 2, August 20-23, 1979.

Hayes-Roth, F., Theory-Driven Learning: Proofs and Refutations as a Basis for Concept Discovery, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Hayes-Roth, F., Learning by Example, Cognitive Psychology and Instruction (Lesgold, A. M., Pellegrino, J. W., Fokkema, S., and Glaser, R., eds.), Plenum, NY, 1978.

Hayes-Roth, F., Patterns of Induction and Associated Knowledge Acquisition Algorithms, Pattern Recognition and Artificial Intelligence (Chen, C. H., ed.), Academic Press, New York, 1976.

Hayes-Roth, F., Representation of Structured Events and Efficient Procedures for Their Recognition, Pattern Recognition 8, 1976, 141-150.

Hayes-Roth, F., Uniform Representations of Structured Patterns and an Algorithm for the Induction of Contingency-Response Rules, Informat. Control 33, 1976, 87-116.

Hayes-Roth, F. and V. R. Lesser, Focus of Attention in the Hearsay-II Speed Understanding System, IJCAI, Vol. 1, August 22-25, 1977, pp. 27-35.

Hayes-Roth, F. and J. McDermott, Knowledge Acquisition from Structural Descriptions, IJCAI, Vol. 1, August 22-25, 1977, pp. 356-362.

Hayes-Roth, F. and J. McDermott, Learning Structured Patterns from Examples, Proc. 3rd Int. Joint Conf. Pattern Recognition, Coronado, CA, 1976, 419-423.

Hayes-Roth, F. and D. Mostow, An Automaticaly Compilable Recognition Network for Structured Patterns, IJCAI, Vol. 1, September 3-8, 1975, pp. 246-252.

Hedrick, C. L., Learning Production Systems from Examples, Artificial Intelligence, Vol. 7, 1976, pp. 21-50.

Hendrick, C. L., A Computer Program to Learn Production Systems Using a Semantic Net, Ph.D. Dissertation, Graduate School of Industrial Adm., Carnegie-Mellon Univ., Pittsburgh, PA, 1974.

Hendrix, G., Expanding the Utility of Semantic Networks Through Partitioning, IJCAI, Vol. 1, September 3-8, 1975, pp. 115-121.

Herman, M., Towards Learning Descriptions of Three-Dimensional Objects, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Hewitt, C., Viewing Control Structures as Patterns of Passing Messages, Artificial Intelligence, Vol. 8, 1977, pp. 323-364.

Hewitt, C., How to Use What You Know, IJCAI, Vol. 1, September 3-8, 1975, pp. 189-198.

Israel, D. J., What's Wrong with Non-Monotonic Logic?, NCAI, August 18-20, 1980, pp. 99-101.

Jouannau, J.-P., G. Guiho and J.-P. Treuil, SISP/1: An Interactive System Able to Synthesize Functions From Examples, IJCAI, Vol. 1, August 22-25, 1977, pp. 412-418.

Kahn, K. M., Making Aesthetic Choices, IJCAI, Vol. 2, August 20-23, 1979.

Kahn, K. and G. A. Gorry, Mechhanizing Temporal Knowledge, Artificial Intelligence, Vol. 9, 1977, pp. 87-110.

Kaminuma T., Conceptual Lattice: A Unified Model for Medical Inference Processes, IJCAI, Vol. 1, August 20-23, 1979.

Kanade, T., Model Repesentations and Control Structures in Image Understanding, IJCAI, Vol. 1, August 22-25, 1977, pp. 1074-1082.

Kawada, T. and S. Amano, Japanese Word Processor, IJCAI, Vol. 2, August 20-23, 1979.

Knapman, J., Some Principles of Artificial Learning that Have Emerged from Examples, IJCAI, Vol. 1, September 3-8, 1975, pp. 253-259.

Konolige, K., An Inference Net Compiler for the PROSPECTOR Rule-Based Consultation System, IJCAI, Vol. 2, August 20-23, 1979.

Konolige, K. and N. J. Nilsson, Multiple-Agent Planning Systems, NCAI, August 18-20, 1980, pp. 138-142.

Kuipers, B., Common-Sense Knowldge of Space: Learning from Experience, IJCAI, Vol. 2, August 20-23, 1979.

Kuipers, B., Modelling Spatial Knowledge, IJCAI, Vol. 1, August 22-25, 1977, pp. 292-298.

Kurokawa, T., Lisp Activities in Japan, IJCAI, Vol. 2, August 20-23, 1979.

Kuzin, E., G. Pozdnyak, and I. Fominykh, Planning the Activity of Robot with Artificial Intelligence, IJCAI, Vol. 1, September 3-8, 1975, pp. 199-205.

Langley, P., Rediscovering Physics with BACON-3, IJCAI, Vol. 2, August 20-23, 1979.

Langley, P. W., BACON: A Production System That Discovers Empirical Laws, IJCAI, Vol. 1, August 22-25, 1977, pp. 344.

Langley, P., G. Bradshaw, and H. A. Simon, Rediscovering Chemistry with BACON.4, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Larson, J. and R. S. Michalski, Inductive Inference of VL Decision Rules, Proc. Workshop Pattern-Directed Inference Systems, SIGART Newsletter 63, 1977, 38-44.

Lauriere, J. L., Toward Efficiency Through Generality, IJCAI, Vol. 2, August 20-23, 1979.

Lebowitz, M., Language and Memory: Generalization as a Part of Understanding, NCAI, August 18-20, 1980, pp. 324-326.

Lehnert, W. G., A Conceptual Theory of Question Answering, IJCAI, Vol. 1, August 22-25, 1977, pp. 158-164.

Leitner, H. H. and M. Freeman, Structured Inheritance Networks and Natural Language Understanding, IJCAI, Vol. 2, August 20-23,

1979.

Lenat, D., BEINGS: Knowledge as Interacting Experts, IJCAI, Vol. 1, September 3-8, 1975, pp. 126-133.

Lenat, D. B., Automated Theory Formation in Mathematics, IJCAI, Vol. 1, August 22-25, 1977, pp. 833-842.

Lenat, D. B., On Automated Scientific Theory Formation: A Case Study Using the AM Program, Machine Intelligence 9, 1978, pp. 251-286.

Lenat, D. B., Computers and Thought Lecture: The Ubiquity of Discovery, IJCAI, Vol. 1, August 22-25, 1977, pp. 1093.

Lenat, D. B., The Ubiquity of Discovery, Artificial Intelligence, Vol. 9, 1977, pp. 257-286.

Lenat, D., AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search, SAIL AIM-286, Stanford Artificial Intelligence Laboratory, Stanford, CA, 1976. Jointly issued as Computer Science Dept. Repot No. STAN-CS-76-570.

Lenat, D. and G. Harris, Designing a Rule System that Searches for Scientific Discoveries, Pattern-Directed Inference Systems.

Lenat, D. B., F. Hayes-Roth and P. Klahr, Cognitive Economy in Artificial Intelligence Systems, IJCAI, Vol. 2, August 20-23, 1979.

Lenat, D. B. and J. McDermott, Less Than General Production System Architectures, IJCAI, Vol. 1, August 22-25, 1977, pp. 928-932.

Lesser, V. R. and L. D. Erman, A Retrospective View of the Hearsay-II Architecture, IJCAI, Vol. 1, August 22-25, 1977, pp. 790-800.

Levesque, H. J. and J. Mylopoulos, An Overview of a Procedural Approach to Semantic Networks, IJCAI, Vol. 1, August 22-25, 1977, pp. 283.

Levi, G. and F. Sirovich, A Problem Reduction Model for Non-Independent Subproblems, IJCAI, Vol. 1, Sepember 3-8, 1975, pp. 340-344.

Loveland, D., Report on Automatic Deduction Symposium, IJCAI, Vol. 1, August 22-25, 1977, pp. 993.

Lugar, G. and G. Goldin, Problem Structure and Problem Solving Behavior, IJCAI, Vol. 1, Sepember 3-8, 1975, pp. 924-931.

Marinov, V., Computer Understanding of Mathematical Proofs, IJCAI, Vol. 1, August 22-25, 1977, pp. 851-857.

Mark, W., Rule-Based Inference in Large Knowledge Bases, NCAI, August 18-20, 1980, pp. 190-194.

Mark, W. S., The Reformulation Approach to Building Expert Systems, IJCAI, Vol. 1, August 22-25, 1977, pp. 329-335.

Marsland, T. A., A Bibliography of Computer Chess, Machine Intelligence 9, 1978, pp. 385-406.

Martin, N., P. Friedland, J. King and M. Stefik, Knowledge Based Management for Experiment Planning Molecular Genetics, IJCAI, Vol. 1, August 22-25, 1977, pp. 882-887.

Mays, E., Failures in Natural Language Systems: Applications to Data Base Query Systems, NCAI, August 18-20, 1980, pp. 327-330.

McCarty, L. T., Some Requirements for a Computer-Based Legal Consultant, NCAI, August 18-20, 1980, pp. 298-300.

McCarthy, J., Epistemological Problems of Artificial Intelligence, IJCAI, Vol. 1, August 22-25, 1977, pp. 1038-1044.

McCorduck, P., M. Minsky, O. Selfridge, and H. A. Simon, History of Artificial Intelligence, IJCAI, Vol. 1, August 22-25, 1977, pp. 951-954.

McCune, B. P., Incremental, Informal Program Acquisition, NCAI, August 18-20, 1980, pp. 71-73.

McDermott, J., R1: An Expert in the Computer Systems Domain, NCAI, August 18-20, 1980, pp. 269-271.

McDermott, J., Learning to Use Analogies, IJCAI, Vol. 1, August 20-23, 1979.

McDermott, D., Vocabularies for Problem Solver State Descriptions, IJCAI, Vol. 1, August 22-25, 1977, pp. 229-234.

McKeown, K. R., Generating Relevant Explanations: Natural Language Responses to Questions About Database Structures, NCAI, August 18-20, 1980, pp. 306-309.

Michie, D., A Theory of Advice, Machine Intelligence 8, 1977, pp. 151-170.

Michalski, R. S., Inductive Learning as Rule-Guided Generalization and Conceptual Simplifiation of Symbolic Descriptions, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Michalski, R. S., A System of Programs for Computer-Aided Induction: A Summary, IJCAI, Vol. 1, August 22-25, 1977, pp. 319-320.

Miller, M. L. and I. P. Goldstein, Structured Planning and Debugging, IJCAI, Vol. 1, August 22-25, 1977, pp. 773-779.

Miller, P., An Adaptive Natural Language System that Listens, Asks, and Learns, IJCAI, Vol. 1, September 3-8, 1975, pp. 406-413.

Minsky, M., Plain Talk About Neurodevelopmental Epistemology, IJCAI, Vol. 1, August 22-25, 1977, pp. 1083.

Mitchell, T. M., An Analysis of Generalization as a Search Problem, IJCA, Vol. 1, August 20-23, 1979.

Mitchell, T. M., P. E. Utgott, R. B. Banerji, Learning Problem-Solving Heuristics by Experimentation, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Mizoguchi, R. and O. Kakusho, Hierarchical Production System, IJCAI, Vol. 2, August 20-23, 1979.

Mizoguchi, F., K. Maruyama, K. Kitazawa, M. Saito, and C. Kulikowski, A Case Study of EXPERT Formalism - An Approach to a Design of Medical Consultation System Through EXPERT Formalisms, IJCAI, Vol. 2, August 20-23, 1979.

Moll, R. and J. W. Ulrich, The Synthesis of Programs by Analogy, IJCAI, Vol. 2, August 20-23, 1979.

Moore, R. C., Reasoning About Knowledge and Action, IJCAI, Vol. 1, August 22-25, 1977, pp. 223-227.

Morgan, C., Automated Hypothesis Generation Using Extended Induction Resolution, IJCAI, Vol. 1, September 3-8, 1975, pp. 351-356.

Mostow, J., Machine-Aided Operationalization of Advice, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Mostow, J. and F. Hayes-Roth, Operationalizing Heuristics: Some AI Metods for Assisting AI Programming, IJCAI, Vol. 2, August 20-23, 1979.

Mylopoulos, J., P. Cohen, A. Borgida, L. Sugar, Semantic Networks and te Generations of Context, IJCAI, Vol. 1, September 3-8, 1975, pp. 134-142.

Nagao, M. and J. Tsujii, S-Net: A Foundation for Knowledge Representation Languages, IJCAI, Vol. 2, August 20-23, 1979.

Neves, D. M., Learning Procedures from Examples, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Nii, H. P. and N. Aiello, AGE (Attempt to Generalize): A Knowledge-Based Program for Building Knowledge-Based Programs, IJCAI, Vol. 2, August 20-23, 1979.

Nishida, F., Y. Fujita, and H. Kusaka, Problem Solving of Elementary Algebra by Hierarchical Abstraction, IJCAI, Vol. 2, August 20-23, 1979.

Novak, G. S., Representations of Knowledge in a Program for Solving Physics Problems, IJCAI, Vol. 1, August 22-25, 1977, pp. 286-291.

Novak, G. S. and A. A. Arays, Research on Expert Problem Solving in Physics, NCAI, August 18-20, 1980, pp. 178-180.

Ogawa, H., H. Nanba, and K. Tanaka, An Active Frame for Knowledge Representation, IJCAI, Vol. 2, August 20-23, 1979.

Oleson, C. E., EXAMINER: A System Using Contextual Knowledge for Analysis of Diagnostic Behavior, IJCAI, Vol. 1, August 22-25, 1977, pp. 814-818.

Pople, H., The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning, IJCAI, Vol. 1, August 22-25, 1977, pp. 1030-1037.

Quinlan, J. R., Inductive Inference as a Tool for the Construction of High-Performance Programs, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Raphael, B., A Computer Program Which "Understands," Proc. Fall Joint Comput. Conf. 26, Spartan Press, Baltimore, MD, 1964, 577-589.

Reder, L. M. and J. R. Anderson, Use of Thematic Information to Speed Search of Semantic Nets, IJCAI, Vol. 2, August 20-23, 1979.

Reinstein, H. C., Problem Solving in Frame-Structured Systems Using Interactive Dialog, NCAI, August 18-20, 1980, pp. 146-147.

Rich, C., H. Shrobe, and R. Waters, Overview of the Programmer's Apprentice, IJCAI, Vol. 2, August 20-23, 1979.

Rich, E., Buildding and Exploiting User Models, IJCAI, Vol. 2, August 20-23, 1979.

Rieger, C., On Organization of Knowledge for Problem Solving and Language Comprehension, Artificial Intelligence, Vol. 7, 1976, pp. 89-128.

Rieger, C. and M. Grinberg, The Declarative Representation and Procedural Simulation of Causality in Pysical Mecanisms, IJCAI, Vol. 1, August 22-25, 1977, pp. 250-256.

Rieger, C. and P. London, Subgoal Protection and Unraveling During Plan Synthesis, IJCAI, Vol. 1, August 22-25, 1977, pp. 487-493.

Rieger, C. and S. Smal, Work Expert Parsing, IJCAI, Vol. 2, August 20-23, 1979.

Rieger, C., Conceptual Overlays: A Mechanism for the Interpretation of Sentence Meaning in Context, IJCAI, Vol. 1, September 3-8, 1975, pp. 143-150.

Rissland, E. L. and E. M. Soloway, Overview of an Example Generation System, NCAI, August 18-20, 1980, pp. 256-258.

Robinson, A. E. and D. E. Wilkins, Representing Knowledge in an Interactive Planner, NCAI, August 18-20, 1980, pp. 148-150.

Rosenfeld, A., J. A. Felman, L. Kanal and P. H. Winston, AI and Pattern Recognition, IJCAI, Vol. 1, August 22-25, 1977, pp. 993.

Rosenberg, S., Reasoning in Incomplete Domains, IJCAI, Vol. 1, August 20-23, 1979.

Rosenberg, S. and B. Roberts, Coreference in a Frame Database, IJCAI, Vol. 2, August 20-23, 1979.

Rychener, M., C. Forgy, P. W. Langley, J. McDermott, A. Newell, and K. Ramakrisna, Problems in Building an Instructable Production System, IJCAI, Vol. 1, August 22-25, 1977, pp. 337.

Rychener, M. D., A Semantic Network of Production Rules in a System for Describing Compuer Structures, IJCAI, Vol. 2, August 20-23, 1979.

Rychener, M. D. and A. Newell, An Instructable Production System: Basic Design Issues, Pattern-Directed Inference Systems.

Sacerdoti, E., The Nonlinear Nature of Plans, IJCAi, Vol. 1, September 3-8, 1975, pp. 206-214.

Samuel, A. L., Some Studies of Machine Learning Using the Game of Checkers, Computers and Though (Feigenbaum, E. A., and Feldman, J., eds.), McGraw-Hill, NY, 1963, 71-105.

Sandewall, E., Some Observations on Conceptual Programming, Machine Intelligence 8, 1977, pp. 223-265.

Schank, R. C., Interestingness: Controlling Inferences, Artificial Intelligence, Vol. 12, No. 3, November 1979, pp. 273-298.

Schank, R., Representation and Understanding of Text, Machine Intelligence 3, 1977, pp. 575-620.

Schank, R. and R. Abelson, Scripts, Plans and Knowledge, IJCAI, Vol. 1, September 3-8, 1975, pp. 151-157.

Schank, R. C., E. Charniak, Y. Wiks, T. Winograd and W. A. Woods, Natural Language Processing, IJCAI, Vol. 1, August 22-25, 1977, pp. 1007-1013.

Schank, R. C. and G. DeJong, Purposive Understanding, Machine Intelligence 9, 1978, pp. 459.

Schank, R. and J. Kolodner, Retrieving Information from an Episodic Memory, or Why Computer's Memories Should be More Like People's, IJCAI, Vol. 2, August 20-23, 1979.

Schank, R. C., M. Lebowitz, and L. Birnbaum, Parsing Directing into Knowledge Structures, IJCAI, Vol. 1, August 20-23, 1979.

Schank, R. C. and W. G. Lehnert, The Conceptual Content of Conversation, IJCAI, Vol. 2, August 20-23, 1979.

Schank, R. C. and M. Selfridge, How to Learn/What to Learn, IJCAI, Vol. 1, August 22-25, 1977, pp. 8-14.

Schmidt, C. F. and N. S. Sridharan, Plan Recognition Using a Hypothesize and Revise Pardigm, IJCAI, Vol. 1, August 22-25, 1977, pp. 480-486.

Schubert, L. K., Problems with Parts, IJCAI, Vol. 1, August 20-23, 1979.

Schubert, L., Extending the Expressive Power of Semantic Networks, IJCAI, Vol. 1, September 3-8, 1975, pp. 158-164.

Schwind, C., Generating Hierarchical Semantic Networks from Natural Language Discourse, IJCAI, Vol. 1, September 3-8, 1975, pp. 429-434.

Self, J. A., Concept Teaching, Artificial Intelligence, Vol. 1, 1977, pp. 197.

Shaw, D., W. Swartout, and C. Green, Inferring LISP Programs from Examples, IJCAI, Vol. 1, September 3-8, 1975, pp. 260-267.

Shrobe, H. E., Dependency Directed Reasoning in the Analysis of Programs Which Modify Complex Data Structures, IJCAI, Vol. 2, August 20-23, 1979.

Siklossy, L. and D. A. Sykes, Automatic Program Synthesis from Example, Proc. 4th Int. Joint Conf. Artificial Intelligence, Tbilisi, USSR, 1975, 268-273.

Siklossy, L. and D. Sykes, Automatic Program Synthesis from Example Problems, IJCAI, Vol. 1, September 3-8, 1975, pp. 268-273.

Simon, H. A., Artificial Intelligence Systems that Understand, IJCAI, Vol. 1, August 22-25, 1977, pp. 1059-1073.

Simon, H. A. and K. Kotovsky, Human Acquisition of Concepts for Sequential Patterns, Psych. Rev. 70, 1963, 534-546.

Simon, H. A. and G. Lea, Problem Solving and Rule Induction: A Unified View, CMU Complx Information Processing Working Paper 227 (revised), Carnegie-Mellon Univ., Pittsburgh, PA, 1963.

Smith, D. E., FOCUSER, A Strategic Interaction Paradigm for Language Acqusition, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Smith, D. E. and J. E. Clayton, A Frame-Based Production System Architecture, NCAI, August 18-20, 1980, pp. 154-156.

Smith, R. G., T. M. Mitchell, R. A. Chestek, and B. G. Buchanan, A Model for Learning Systems, IJCAI, Vol. 1, August 22-25, 1977, pp. 338-343.

Smith, R. L., Jr., Modelling Student Acquisition of Problem-Solving Skills: The Student's Interpretation of the Teaching Environement, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Solomonoff, R., Inductive Inference Theory - A Unified Approach to Problems in Pattern Recognition and Artificial Intelligence, IJCAI, Vol. 1, September 3-8, 1975, pp. 274-280.

Soloway, E. M. and E. M. Riseman, A Common Sense Approach to Learning, Proc. Workshop Pattern-Directed Inference Systems, SIGART Newsletter 63, 1977, 49-55.

Soloway, E. M. and E. M. Riseman, Levels of Pattern Description in Learning, IJCAI, Vol. 1, August 22-25, 1977, pp. 801-811.

Sondheimer, N. K., Towards a Combined Representation for Spacial and Temporal Reference, IJCAI, Vol. 1, August 22-25, 1977, pp. 281-282.

Sridharan, N. S. and F. Hawrisik, Representation of Actions that have Side-Effects, IJCAi, Vol. 1, August 22-25, 1977, pp. 265-266.

Srinivesan, C. V., Knowledge Based Learning Systems, An Example, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Stallman, R. M. and G. J. Sussman, Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis, Artificial Intelligence, Vol. 9, 1977, pp. 135-196.

Stefik, M., An Examination of a Frame-Structured Representation System, IJCAI, Vol. 2, August 20-23, 1979.

Steinberg, L., Question Ordering in a Mixed Initiative Program Specification Dialogue, NCAI, August 18-20, 1980, pp. 61-63.

Stepp, R., Learning from Observation: Experiments in Conceptual Clustering, Workshop on Current Developments in Machine Learning - Part I, Pittsburgh, PA, July 16-18, 1980.

Sussman, G. J., Electrical Design: A Problem for Artificial Intelligence Research, IJCAI, Vol. 1, August 22-25, 1977, pp. 894-900.

Sussman, G. J., A Computational Model of Skill Acquisition, American Elsevier, NY, 1975.

Swartout, W. R., A Digitalis Therapy Advisor with Explanation, IJCAI, Vol. 1, August 22-25, 1977, pp. 819-825.

Taimoto, S. L., Inductive Learning of Categories from Examples Using Minimum Cost Representations, IJCAI, Vol. 2, August 20-23, 1979.

Tate, A., Generating Project Networks, IJCAI, Vol. 1, August 22-25, 1977, pp. 888-893.

Tate, A., Interacting Goals and Their Use, IJCAI, Vol. 1, September 3-8, 1975, pp. 215.

Tennant, H., Experience with the Evaluation of Natural Language Question Answers, IJCAI, Vol. 2, August 20-23, 1979.

Thompson, A. M., Network Truth Maintenance for Deduction and Modelling, IJCAI, Vol. 2, August 20-23, 1979.

Thorndyke, P. W., Heuristics for Knowledge Acquisition from Maps, IJCAI, Vol. 2, August 20-23, 1979.

Thorndyke, P. W., Knowledge Transfer in Learning from Texts, Cognitive Psychology and Instruction (Lesgold, A. M., Pellegrino, J. W. , Fokkema, S., and Glaser, R., eds.), Plenum, NY, 1978.

Tikhomirov, O., Philosophical and Psychological Problems of Artificial Intelligence, IJCAI, Vol. 1, September 3-8, 1975, pp. 932-937.

Towster, E., Concept Decomposition as a Method of Concept Formation, IJCAI, Vol. 1, August 22-25, 1977, pp. 347.

Trigoboff, M. and C. A. Kulikowski, IRIS: A System for te Propagation of Inferences in a Semantic Net, IJCAI, Vol. 1, August 22-25, 1977, pp. 274-280.

van Melle, W., A Domain-Independent Production-Rule System for Consultation Programs, IJCAI, Vol. 2, August 20-23, 1979.

Vere, S. A., Multilevel Counterfactuals for Generalizations of Relational Concepts and Productions, Artificial Intelligence, Vol. 14, No. 2, September 1980, pp. 139-164.

Vere, S. A., Relational Production Systems, Artificial Intelligence, Vol. 8, 1977, pp. 47-68.

Vere, S. A., Induction of Relational Production in the Presence of Background Information, IJCAI, Vol. 1, August 22-25, 1977, pp. 349-355.

Vere, S. A., Induction of Relational Productions in the Presence of Background Information, Proc. 5th Int. Conf. Artificial Intelligence, Cambridge, MA, 1977, 349-355.

Vere, S., Induction of Concepts in the Predicate Calculus, IJCAI, Vol. 1, September 3-8, 1975, pp. 281-287.

Waldinger, R., Achieving Several Goals Simultaneously, Machine Intelligence 8, 1977, pp. 94-138.

Waldinger, R. and Z. Manna,, Knowledge and Reasoning in Program Synthesis, IJCAI, Vol. 1, September 3-8, 1975, pp. 288-295.

Walker, D. E., W. H. Paxton, B. J. Grosz, G. G. Hendrix, A. E. Robinson, J. J. Robinson and J. Slocum, Procedures for Integrating Knowledge in a Speech Understanding System, IJCAI, Vol. 1, August 22-25, 1977, pp. 36-42.

Waltz, D., Natural Language Access to a Large Data Base: An Engineering Approach, IJCAI, Vol. 1, September 3-8, 1975, pp. 868-872.

Waterman, D. A., Rule-Directed Interactive Transaction Agents: An Approach to Knowledge Acquisition, R-2171-ARPA, The Rand Corporation, Santa Monica, CA, 1977.

Waterman, D. A., A Rule-Based Approach to Knowledge Acquisition for Man- Machine Interface Programs, P-5895, The Rand Corporation, Santa Monica, CA, 1977.

Waterman, D. A., Serial Pattern Acquisition: A Production System Approach, Pattern Recognition and Artificial Intelligence (Chen, C. H., ed.), Academic Press, NY, 1976, 529-553.

Waterman, D., Adaptive Production Systems, IJCAI, Vol. 1, September 3-8, 1975, pp. 296-303.

Waterman, D. A., Adaptive Production Systems, Proc. 4th Int. Joint Conf. Artificial Intelligence, Tbilisi, USSR, 1975, 296-303.

Waterman, D. A., Generalization Learning Techniques for Automating the Learning of Heuristics, Artificial Intelligence, 1, 1970, 121-170.

Waterman, D. A., Machine Learning of Heuristics, Ph.D. Disseration, Comput. Sci. Dept., Stanford Univ., Stanford, CA, 1968.

Waterman, D. A. and M. Peterson, Rule-Based Models of Legal Expertise, NCAI, August 18-20, 1980, pp. 272-275.

Waters, R. C., A Method for Automatically Analyzing Programs, IJCAI, Vol. 2, August 20-23, 1979.

Weinzweig, M., Association Memory Model Using the Notion of "Importance," IJCAI, Vol. 1, September 3-8, 1975, pp. 165.

Weiss, R. and C. Kulikowski, EXPERT: A System for Developing Consultation Models, IJCAI, Vol. 2, August 20-23, 1979.

Weiss, S. M., C. A. Kulikowski, and B. Nudel, Learning Production Rules for Consultation Systems, IJCAI, Vol. 2, August 20-23, 1979.

Weiss, S. M., C. A. Kulikowski and A. Safir, A Model-Based Consultation System for the Long-Term Management of Glaucoma, IJCAI, Vol. 1, August 22-25, 1977, pp. 826-832.

Wesson, R. B., Planning in the World of the Air Traffic Controller, IJCAI, Vol. 1, August 22-25, 1977, pp. 473-479.

Whitehill, S. B., Self-Correcting Generalization, NCAI, August 18-20, 1980, pp. 240-242.

Wilensky, R., Meta-Planning, NCAI, August 18-20, 1980, pp. 334-336.

Wilks, Y. and J. Bien, Speech Acts and Multiple Environments, IJCAI, Vol. 2, August 20-23, 1979.

Winston, P. H., Learning by Creating and Justifying Transfer Frames, Artificial Intelligence, Vol. 10, No. 2, April 1978, pp. 147-172.

Winston, P. H., Learning Structural Descriptions from Examples, The Psychology of Computer Vision (Winston, P. H., ed.), McGraw-Hill, NY, 1975.

Wood, R. J., A Program Model and Knowledge Base for Computer Aided Program Synthesis, NCAI, August 18-20, 1980, pp. 77-78.

Yosida, S., Hierarchical Concepts Structure for Natural Language Understanding System, IJCAI, Vol. 2, August 20-23, 1979.

Young, R. M., G. D. Plotkin and R. F. Linz, Analysis of an Extended Concept-Learning Tasks, IJCAI, Vol. 1, August22-25, 1977, pp. 348.

Zadeh, L. A., PRUF - A Language for the Representation of Meaning in Natural Languages, IJCAI, Vol. 1, August 22-25, 1977, pp. 918.